الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي

جامعة الإخوة منتوري قسنطينة 1
كلية العلوم الدقيقة
قسم الفيزياء

الرقم التسلسلي: .....
السلسلة : .....

أطروحة

مقدمة لنيل شهادة دكتوراه في العلوم

تخصص: الفيزياء النظرية

بعنوان:

المشي العشوائي الكمي و خاصية الانتقال في بُنَى الجرافين

من طرف:

بوقرورة حمزة

تناقش يوم: ../../ 201.

أمام اللجنة:

الرئيس:	نور الدين مباركي	أستاذ	جامعة الإخوة منتوري قسنطينة 1
المشرف:	حبيب عيساوي	أستاذ	جامعة الإخوة منتوري قسنطينة 1
الأعضاء:	عاشور بن سلامة	أستاذ	جامعة الإخوة منتوري قسنطينة 1
	منير بوساهل	أستاذ	جامعة محمد بوضياف - المسيلة
	الدراجي بهلول	أستاذ	جامعة الحاج لخضر باتنة 1
	مصطفى مومني	أستاذ محاضر "أ"	جامعة محمد خيضر بسكرة

جدول المحتويات:

بسم الله الرحمن الرحيم

إن الحمد لله نحمده ونستعينه ونستغفره ونعوذ بالله من شرور أنفسنا ومن سيئات أعمالنا من يهده الله فلا مضل له ومن يضلل فلا هادي له وأشهد أن لا إله إلا الله وحده لا شريك له وأشهد أن محمدا عبده ورسوله.

{يَا أَيُّهَا الَّذِينَ آمَنُوا اتَّقُوا اللَّهَ حَقَّ تُقَاتِهِ وَلا تَمُوتُنَّ إِلَّا وَأَنْتُمْ مُسْلِمُونَ}.

{يَا أَيُّهَا النَّاسُ اتَّقُوا رَبَّكُمُ الَّذِي خَلَقَكُمْ مِنْ نَفْسٍ وَاحِدَةٍ وَخَلَقَ مِنْهَا زَوْجَهَا وَبَثَّ مِنْهُمَا رِجَالاً كَثِيراً وَنِسَاءً وَاتَّقُوا اللَّهَ الَّذِي تَسَاءَلُونَ بِهِ وَالْأَرْحَامَ إِنَّ اللَّهَ كَانَ عَلَيْكُمْ رَقِيباً}

{يَا أَيُّهَا الَّذِينَ آمَنُوا اتَّقُوا اللَّهَ وَقُولُوا قَوْلاً سَدِيداً، يُصْلِحْ لَكُمْ أَعْمَالَكُمْ وَيَغْفِرْ لَكُمْ ذُنُوبَكُمْ وَمَنْ يُطِعِ اللَّهَ وَرَسُولَهُ فَقَدْ فَازَ فَوْزاً عَظِيماً} أما بعد.

تـــشـــكـــرات

أتقدم بجزيل الشكر و العرفان لكل من ساهم من قريب أو من بعيد في هذا العمل المتواضع.
و أخص بالذكر:
الأستاذ: ح. عيساوي الذي رافقني و سَّهل و أشرف على عملي
الأستاذة: ف. كيندون (V. Kendon) التي استضافتني و ساهمت في تهذيب و إثراء أفكاري
السادة: ن. مباركي، ع. بن سلامة، م. بوساهل، د. بهلول و م. مومني.
على قبولهم مناقشة أطروحتي
وزارة التعليم العالي و البحث العلمي التي مولت ابتعاثي
جامعة تبسة التي وافقت على انتدابي
زميلي الأستاذ أ. حوحو الذي استفدت من تجربته في تيسير إجراءات سفري و ابتعاثي
و مراجعته القيمة في قرآءة الأطروحة و تحديد عثراتها النحوية و اللغوية
كما لا أنسى الأستاذ ع. بن سلامة الذي كانت له مساهمة في مراجعة الأطروحة يشكر عليها كثيرا
و كل من ساهم بالنصيحة و الانشغال
فهم مشكورين مأجورين بإذن الله

إهـــــــــــداء

أهدي هذا الجهد المتواضع إلى:

الوالدين الكريمين اللذين ساندني و باهتمام

و لعمي الذي كان يتابعني و بانشغال

و لزوجتي الصبورة

و لإبنتيّ اللعوبتيّن

و لإخوتي و جميع أهل بيتي

و لجميع جيراني و أصحابي

و للغتي

لغة القرآن و السنّة و الإسلام، لغة الحكمة و الفصاحة و البيان، لغة الخير و البلاغة و طلاقة اللسان

و لكل من كان له فضل علي

الفصل الأول:

تقديم الموضوع

إن فكرة المشي العشوائي الكمي[1] ما هي في الحقيقة إلا النسخة الكمية لفكرة المشي العشوائي الكلاسيكي الذي يصنف ضمن الظواهر الإحصائية الاحتمالية، و يُعرَّفُ اختصارا من اسمه على شكل نموذج رياضياتي يدرس السلوك العام و خصائصه وفق مبادئ الفيزياء الكلاسيكية لتطور الزمني لموضع نقطة متحركة (أو حالة فيزيائية) من خلال مجموعة من الخطوات (أو التحولات) العشوائية المتماثلة و المتكررة، عبر مسارات و اتجاهات ممكنة فقط تحددها البنية المميزة لما يسمى بالمِخْطَاطْ (graph) الذي تتطور وفقه، سواء في حالة الزمن المستمر أو الغير مستمر[2،3].

عموما يعرف أي مخطاط (يرمز له بالحرف G) على أنه مجموعة معينة من العقد عددها الكلي N، تتصل فيما بينها على شكل ثنائيات. عن طريق وصلات ممكنة فقط بحسب بنية كل مخطاط، عددها الكلي هو E، تُعَلَّمُ كل عقدة بعدد صحيح n، و منه يمكن أن نُعِلِّم كل وصلة ممكنة بين عقدتين مختلفتين n و $\grave{n}$ بالثنائية $(\grave{n}، n)$. إذا كان عدد الوصلات التي تنطلق من كل عقدة (يرمز عادة لهذا العدد بـ d) هو نفسه لجميع العقد يسمى هذا المخطاط بالمنتظم. و إذا كانت الحركة أو الخطوات عبر كل وصلة بين أي عقدتين متواصلتين تتم في الاتجاهين فيسمى هذا المخطاط بغير الموجه.

إن للمشي العشوائي الكلاسيكي استعمالات كثيرة و ناجحة و في عدة ميادين متنوعة: كالتنبؤات الاقتصادية أو الاستقرءات المالية، دراسة السلوك العام لكثير من الأنظمة الديناميكية الإحصائية أو العشوائية، تطوير صنف معين من الخوارزميات العشوائية التي تفيد في حل صنف معين من معضلات الحوسبة الكلاسيكية ...إلخ[4-6].

في العقدين الأخيرين. بدأ الاهتمام يتزايد كثيرا بالمشي الكمي مقارنتة مع المشي الكلاسيكي. و ذلك مواكبتة مع بروز فكرة الحوسبة الكمية نظريا و تطبيقيا و ما يمكن أن تتيحه من استعمالات ثورية

من جهة[7] و ظهور بعض النتائج المشجعة و الهامة لمشي الكمي مقارنتة بالمشي الكلاسيكي من جهة أخرى. هذا كله عزز من توظيف نتائج المشي الكمي في عديد من المواضيع النظرية و التطبيقية [8، 9]كالخوارزميات الكمية البحثية (quantum search algorithems)، الشبكات البصرية الكمية (quantum optical lattice)، التجاويف الالكترو- ديناميكية الكمية (cavity quantum electrodynamics)، الأيونات المحتجزة ضمن الشبكات البصرية (trapped ions in an optical lattice) [10-13، 14]...إلخ.

إن لنوعية بنية المخطاط المختار في المشي العشوائي سواء الكلاسيكي أو الكمي دور مهم في دراسة ما مدى فعالية النماذج الكمية مقارنة بالنماذج الكلاسيكية. و قد جُربت عدة كميات لاختبار هذه الفعالية مثل: زمن الاصطدام (hitting time)، زمن الاستقرار أو التمازج (mixing time)، الموضع المتوسط (average position)، موضع القمة (max position)، خاصية الانتقال (transport property)، الانتقال المثالي للحالة (perfect state transfer)...إلخ [8]. كما أُقترحت بنى أو مخاطيط كثيرة فمنها من كانت نسختها الكمية أسرع من الكلاسيكية أو العكس. و نذكر مثلا من بين أشهر هذه البنى و المخاطيط التي درست لحد الآن: الخط البسيط، الحلقة، المكعب البسيط و المكعب الفائق. الشبكة الكارتيزية الثنائية المستوية أو الثلاثية الفراغية، الشبكات ذات البنية النجمية، الشبكات ذات البنية الشجيرية الأحادية و الثنائية، الشبكات ذات بنى الهندسة التكسيرية و التكرارية...إلخ. حاليا، تعد بنية المخطاط "المكعب الفائق" (hypercube) [15، 16] و المخطاط "التفرع- الشجيري- الثنائي- الملتصق" (glued double trees) [17] من بين أنجح الأعمال التي برهنت على سرعة و كفاءة المشي الكمي مقارنتة بالكلاسيكي و التي شجعت على دراسات و أعمال عدة قامت في مجملها بعرض أفكار جديدة في كيفية دراسة و اختبار كفاءة المشي الكمي مقارنتة بالكلاسيكي نذكر منها: خاصية الانتقال الكمي عبر القنوات الخطية باستعمال السبين أو ما يعرف بـ quantum transport on spin chains [18-20] و أهمية توظيفها كأسلاك بين مختلف أجزاء الأجهزة الكمية سواء للتوصيل أو لتنفيذ العمليات الحسابية الكمية، ما يعرف بـ Anderson localisation [21-23] الذي سلط الضوء على أهمية التحكم في مختلف

الثوابت الذاتية أو الداخلية للمشي الكمي للوصول إلى أحسن معدل انتقال أو أحسن النتائج بصفة عامة [24، 25، 26]، أهمية التناظر في البنى و المخاطيط التي يُدْرَسُ من خلالها كفاءة المشي الكمي [27، 28] و كيفية توظيفه لإيجاد المِخْطَاط أو المَخَاطِيط المثلى للمشي العشوائي. أين تكون نسخته الكمية أسرع و أكفأ من نسخته الكلاسيكية. و هناك حتى من الأعمال التي وظفت ميزات إضافية للمشي في حد ذاته: كفكرة عدة مشائين عشوائيا في آن واحد (multi-walker) [29] و كيف سيتفاعلون في ما بينهم [30]، توظيف خاصية التشابك الكمي (entanglement) بين الحالات الابتدائية لعدة مشائين [31]. استعمال فكرة القياس الضعيف (weak measurement) أثناء التطور الزمني للمشي الكمي لدراسة و فهم موضوع فك الارتباط (decoherence) و كيفية التحول أو الانتقال من المشي الكمي إلى الكلاسيكي بوجود عوامل تأثير المحيط الخارجي [32، 33-35] ...إلخ

إن تناظر بنية المخاطيط لطالما أعتبر عاملا جد مهم، حتى أن هناك من علل به أسباب نجاح مخطاط "المكعب الفائق" و "التفرع الشجيري الثنائي الملتصق" [1]. و بالتالي أصبح مرشحا فوق العادة للعب دور مميز في ايجاد البنى التي تتميز بمشي كمي أسرع من المشي الكلاسيكي [24، 28]. من هنا بدأت الفكرة (أو لنقل أمسكنا بطرف الخيط) حيث لم نجد أي من الأعمال قد درست بنى ذات تناظر كروي رغم أن التناظر الكروي يعتبر هو التناظر ذو الأعلى درجة وفق الهندسة الإقليدية. و هذا التناظر الكروي في البداية قادنا إلى اقتراح مخطاط بنيته العامة كروية و تفاصيله الجزئية تشبه تماما تفاصيل خطوط الطول و دوائر العرض الوهمية التي تحدد تقاطعاتها الأماكن على سطح الكرة الأرضية. لكن مع بروز مشكل القطب الشمالي و الجنوبي أين تتقاطع خطوط الطول بشكل مغاير و أكبر من بقية الأماكن (أي مخطاط غير منتظم) جعلنا نغير نوعا ما من بنية المخطاط إلى البنية الكروية لجزيء الكربون C_{60} أين تفاصيله هي تماما تفصيل كرة القدم التي نعرفها. و كل ذرة تخرج منها ثلاث وصلات فقط (أي مخطاط منتظم حيث $d=3$). و بعد النتائج الأولية الجيدة التي تحصلنا عليها. شجعتنا للمضي قدما نحو بنى أخرى كأنابيب النانو المختلفة (zigzag و armchair) و ذلك لتناظرها الأسطواني. هذا من جهة، و من جهة أخرى ارتباط هذه البنى التي اقترحناها بمجال بحث

جد مهم و جد نشط كـ: "بنى النانو" و "النانو تكنولوجي" و ما يشكله من ثقل بحثي في وقتنا الحالي [36-46] هو ما شجعنا أكثر في المضي قدما في اختبار هذه البنى التي اقترحناها. مؤخرا، هناك بعض الأعمال التي تطرقت لاختبار المشي الكمي المستمر، وفق شبكات بنيتها مثل بنية أوراق الجرافين و لكن بطريقة أخرى أهدافها مغايرة للأهداف التي قمنا بها في هذه الدراسة (دراسات تحليلية تساعد في إيجاد خوارزميات كمية بحثية [47] و طرق و أهداف أخرى [48-50]).

إن الدراسة التحليلية لمشي الكمي عبر مختلف البنى تعتبر تحديا رياضياتيا حقيقيا. و تتطلب مخاطيط ذات بنية متجانسة [20]. فحتى أبسط البنى التي درست تحليليا كالحلقة [51] أو المكعب الفائق [52] أو الشبكة الكارتيزية [53-55] كانت مهمّة ليست بالسهلة. و تطلبت جهدا رياضياتيا معتبرا للوصول الى نتائجها. كما أن الدراسة التحليلية لما يسمى بالانتقال المثالي للحالة (perfect state transfer) عبر البنى الصغيرة أين يتم اختبار إمكانية الوصول لمكان موضع الهدف نفس الحالة الابتدائية بالضبط التي انطلقت من موضع الانطلاق كللت بحلول تحليلية و لكن من أجل عدد محدود فقط من الحالات [56]، أمّا في حالة دراسة البنى التي تستعمل تقنية توليف بنية كبيرة انطلاقا من تركيب مجموعة من البنى الصغيرة [57] حيث تكون حلولها التحليلية هي الاخرى عبارة عن تركيبة لحلول كل هذه البنى الصغيرة حتى هي تعتبر تقنية لها حدود و لا يمكن استعمالها في جميع الحالات. و بالتالي في الحالة العامة، تبقى الدراسات الرقمية هي الخيار الأفضل في التعامل مع أي بنية مهما كان نوعها و تعقيدها و تشعبها.

إن الدراسة التي سنقدمها هنا هي دراسة رقمية[1] للمشي العشوائي غير المستمر. سواء الكلاسيكي أو الكمي عبر بنية مخاطيط عدّة منتظمة غير موجهة كبنى الجرافين. و سنركز على اختبار خاصية الانتقال و كيفية توظيف التناظر في الحالات الإبتدائية من أجل الوصول الى أحسن النتائج. في هذه الأطروحة و بحكم أن الموضوع جديد على الساحة المحلية حسب ظني حاولت أن أستفيض في المقدمات لتكون واضحتا و سهلة الفهم من القراءة الأولى. فحررت فصلين، الأول بعنوان المشي العشوائي الكلاسيكي،

[1] لا تتطلب تجهيزات خاصة. حيث بعض من الدقائق تكون كافية للحصول على النتائج النهائية. و قد استعملنا لغة البرمجة Python و لواحقه [58-61].

نُعَرف فيه المشي الكلاسيكي غير المستمر، و نموذجه الرياضياتي في الحالة العامة لاستخلاص معادلاته. ثم نطبق هذا النموذج على أحد أشهر البنى أو المخاطيط مثل الخط المستمر و الحلقة المنتهية. لنحلل في الأخير النتائج و نستخلص الخصائص. أمّا الثاني فهو بعنوان المشي العشوائي الكمي، استعرضنا فيه النسخة الكمية للمشي العشوائي و كيف يمكننا أن نميز السلوك الكمي عن الكلاسيكي عن طريق عملية القياس و ما يمكن أن يحدثه إذا طبق من عدمه. استعملت أبسط مثال لبناء النسخة أو النموذج الكمي مع التفصيل في جميع المراحل تقريبا. توقفت قليلا عند الاستثناءات و التنبيهات التي من شأنها أن تساعد على الإحاطة بالموضوع بشكل حسن و مقبول. طبقنا المشي الكمي على بعض الأمثلة البسيطة و استعرضنا نتائجها و استخلصنا مميزاتها خاصة التي تغاير النسخة الكلاسيكية. في الفصل الثالث دخلنا في صلب موضوع الاطروحة، و قدمنا فيه بنى الجرافين المختلفة التي تعرضنا لها بالدراسة. و خاصة التعليق و التنبيه على مميزاتها التناظرية التي سنحتاجها في تطبيق نماذج المشي العشوائي الكلاسيكي و الكمي. بعد تطبيق هذه النماذج بالاستعانة بالبرامج الحاسوبية من أجل عدد كبير من الخطوات تحصلنا على النتائج و قارناها مع الحلقة من نفس الحجم لنجد و نستخلص النتائج و نعلق عليها. في الفصل الرابع و الأخير لخصنا نتائج الاطروحة بصفة عامة ثم ركزنا بعدها على مناقشة أهميتها، أولا عن طريق ربطها بنتائج مواضيع أخرى و ما يمكن ان تقدمه كخيرات إضافية مساعدة لتحسين نتائج هذه المواضيع، ثانيا عن طريق تقديم شبه أفكار على شكل أسئلة مفتوحة ربما يمكن أن تكون مواضيع بحث في الأعمال القادمة كامتدادات لما قدم في هذه الأطروحة.

الفصل الثاني:

المشي العشوائي الكلاسيكي

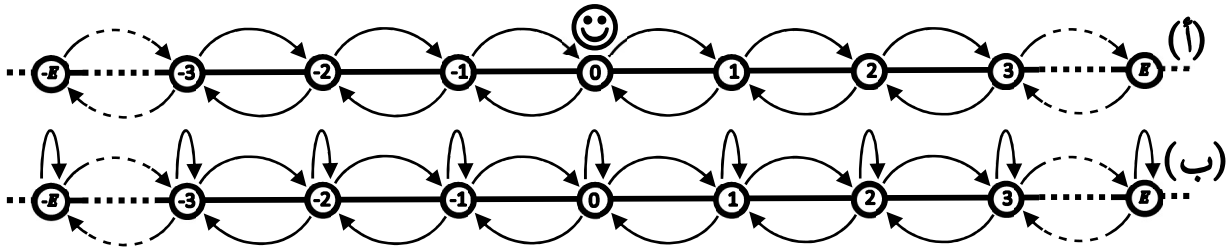
سنحاول في هذا الفصل إعطاء نظرة مفصلة، عن فكرة المشي العشوائي الكلاسيكي غير المستمر وفق البنى المنتظمة. سنستعرض كيفية بناء النموذج الرياضياتي الذي يصف هذا النوع من الحركة العشوائية من الصفر لنصل في الأخير إلى إيجاد المعادلة الزمنية لتطور توزع الاحتمال الكلي للمشي العشوائي وفق أي بنية مخطاط منتظم مهما كان نوعه و شكله (أي الحالة العامة). بعدها سنطبق معادلات هذا النموذج على حالتين خاصتين شهيرتين و هما الخط غير المنتهي أو المستمر و الحلقة المنتهية. سنستعرض المعادلات ثم نحسب تطور توزع الاحتمال الكلي من أجل الخطوات الأولى و بعدها سنتكلم عن الأمور التقنية و كيفية تحرير البرنامج الحاسوبي الذي يمكنه حساب تطور توزع الاحتمال الكلي من أجل عدد كبير من الخطوات. أخيرا، سنستعرض النتائج في شكل رسومات بيانية لنحللها و نستخلص منها أهم مميزات هذا النوع من الحركة العشوائية وفق الخط و الحلقة.

بعدها سنتطرق إلى عرض الكيفية العامة في المقارنة بين مختلف البنى و المخاطيط و ذلك بتقديم فكرة المستويات و البنى المكافئة الناجمة عنها. لنقارن بعدها مباشرة بين الخط المستمر و الحلقة اللذين درسناهما في بداية الفصل. و لكن هذه المرة بتقديم و اختبار خاصية جديدة هي خاصية إنتقال أو تنقل الاحتمال الكلي بين موضعين نختارهما، هما موضع الانطلاق و موضع الوصول. سنقدم شروحات عن فكرة خاصية الانتقال و كفية حسابه كمياً باستعمال المعادلات الزمنية لتطور توزع الاحتمال الكل، ثم تنبيهات تقنية عن كيفية بناء البرنامج الحاسوبي الذي سيقوم بتنفيذ الحسابات من أجل عدد كبير نسبيا من الخطوات. بعد الحصول على النتائج البيانية سنقوم بتحليها لنستنتج أي المخاطيط أفضل في نقل الاحتمال الكلي من الآخر.

في ما يخص المراجع أنصح بالتدرج لمن لا يريد التعمق كثيرا في المراجع على النحو التالي: للحصول على ما قل و دل [2، 3، 8]، ثم لتعمق و التوسع أكثر [4-6].

1.2 بناء النموذج في الحالة العامة (البنى المنتظمة):

لتكن لدينا نقطة مجردة أو شخص عادي يمشي عشوائيا وفق بنية أو مِخْطَاطٍ مُنْتَظِمْ، يتكون من N عقدة. كل عقدة n تنطلق منها وَصْلَتَينِ ترتبط بالعقدتين التي تليها و التي تسبقها عن طريق رابطين ممكنين فقط، معطية شكل خط مستمر (ليس شرط أن يكون مستقيم) يربط بينها، انطلاقا من اختيار مبدأ مناسب لزمان و المكان؛ و لتكن العقدة $n=0$ مثلا كما هو موضح في الوثيقة 01-2. أين سيكون احتمال تواجد الشخص معدوم في جميع العقد باستثناء العقدة 0 أين تكون قيمة الاحتمال فيها هي الاحتمال الكلي المساوي للوحدة "1". سنترك الشخص يمشي عشوائيا ضمن هذا المخطاط حيث قبل كل خطوة سَيَخْطُوهَا، سَيَعْمِدُ إلى رمي قطعة أو عُمْلَةٍ نقدية ذات وجهين (أو زهرة نرد ذات ثلاث وجوه مثلاً)، و حسب نتيجة رمي العملة النقدية، سيتحرك وفق اتجاهيين محتملين و ممكنين فقط في المشي (أو ثلاث)، بخطوة تزيحه نحو اليمين أو بخطوة تزيحه نحو اليسار (أو بشبه خطوة تبقيه في مكانه. سنسميها مثلاً خطوة "استراحة") و باحتمال مُتَسَاوٍ لكل اتجاه، قيمته هي مقلوب عدد الاتجاهات المحتملة او الممكنة "d"، أي: $\frac{1}{\text{الإتجاهات الممكنة}} = \frac{1}{d}$. (يسمى عادة هذا الاحتمال باحتمال العُمْلَة "$coin$"). لحظة بعد لحظة، و خطوة بعد خطوة، ستتكر هذه العملية مرار و تكرار و بشكل متماثل و عشوائي، معطية مع الوقت، توزعا جديدا لاحتمال تواجد الشخص على مستوى كل العقد المشكلة للمخطاط.



الوثيقة 01-2: مخطاط مُنْتَظِمْ يتألف من $N = 2E + 1$ عقدة، ترتبط فيما بينها على هيئة خط مستمر (و لكن طرفيه متصلين للحصول على بنية مُنْتَظِمْ) حيث المشي العشوائي يمكن أن يتم انطلاقا من كل عقدة وفق: (**أ**) اتجاهين ممكنين: "يمينا أو يسارا". (**ب**) ثلاثة اتجاهات ممكنة: "يمينا، يسارا أو استراحة".

لدراسة مميزات و خصائص هذا النوع من المشي العشوائي و الاحتمالي يمكننا بناء نموذج رياضياتي يصف و يحاكي هذا المشي، و ذلك بالاعتماد على خصائص الجداء المصفوفي، حيث في البداية يمكننا ان نعبر عن المخطاط المدروس بفضاء شعاعي بعده هو عدد عقد المخطاط N. و بالتالي فكل عقدة n

من المخطاط سنعبر عنها بأحد أشعة وِحْدَةِ هذا الفضاء الشعاعي التي تحتوي على N مركبة، كلها معدومة باستثناء المركبة التي توافق رتبة شعاع الوحدة المَعْني بالأمر ضمن ترتيب كل أشعة الوحدة المتبقة (أي، رتبة العقدة n المَعْنِيَةِ ضمن ترتيب كل العقد المتبقية)، و المختار حسب الحاجة و حسب الحالة المدروسة. فمثلا في حالة الوثيقة 01-2، سواء في حالة المخطاط (أ) أو (ب) يمكن أن نختار التمثيل الشعاعي للعُقَدِ E,⋯, 1, 0, 1-,..., E- وفق الترتيب التالي:

$$|E\rangle=\begin{pmatrix}0\\0\\0\\0\\\vdots\\1\end{pmatrix},\ldots,|1\rangle=\begin{pmatrix}0\\0\\0\\1\\0\\\vdots\\0\end{pmatrix},|0\rangle=\begin{pmatrix}0\\0\\\vdots\\0\\1\\0\\\vdots\\0\end{pmatrix},|-1\rangle=\begin{pmatrix}0\\\vdots\\0\\1\\0\\0\\\vdots\\0\end{pmatrix},\ldots,|-E\rangle=\begin{pmatrix}1\\\vdots\\0\\0\\0\\\vdots\\0\end{pmatrix}.$$

و انطلاقا من هذا التمثيل الأخير، يمكننا في الحالة العامة أن نعبر شعاعيا عن كل خطوة بالجداء المصفوفي بين شعاع سطر يمينا، يمثل العقدة n التي تبدأ منها الخطوة. مضروباً في شعاع عمود يساراً، يمثل العقدة $\grave{n}$ التي تنتهي إليها الخطوة (و ذلك لوجود وصلة بينهما). أي:

$$|\grave{n}\rangle\langle n| \qquad (2\text{-}01)$$

عادة ما يُعْمَدُ إلى منهجة او تقنين العلاقة التي تربط بين رتبة العقد n التي تنطلق منها الخطوة و رتبة العقد $\grave{n}$ التي تنتهي إليها، و ذلك بحسب تَوزيعٍ و ترميزٍ ممنهج يساعد و يسهل في الدراسات التحليلية خاصة، حيث تكون رتبة العقد $\grave{n}$ التي تنتهي إليها الخطوة هي عبارة عن دالة بالنسبة لرتبة لعقد n التي تنطلق منها الخطوة أي $\grave{n}=f(n)$. و بالتالي في الحالة العامة يمكن التعبير عن جميع الخطوات الممكنة بمصفوفة واحدة تحتوي على جميع الخطوات، عبارتها الشعاعية هي:

$$A=\sum_{n=-E}^{E}|f(n)\rangle\langle n| \qquad (2\text{-}02)$$

سنسمي هذه المصفوفة بمصفوفة الخطوات، عناصرها A_{ij}، تأخذ القيمتين الصفر أو الواحد فقط، معبرة بذلك عن إمكانية حدوث الخطوة و جهتها من عدمها (من العقدة j نحو العقدة i). و للتعبير عن كل خطوة و احتمالها، يكفي أن ندخل احتمال العملة على مصفوفة الخطوات للحصول على ما يسمونه بـ: مصفوفة الانتقال M، حيث:

$$A_{ij}/d = M_{ij} \qquad , \qquad \sum_{i=-E}^{E} M_{ij} = 1 \tag{2-03}$$

بحكم الكتابة المصفوفية لمصفوفة الانتقال M سنحتاج للتعبير عن احتمال جميع العقد بشعاع، نسميه شعاع الاحتمال الكلي$P(T)$، مركباته $P_n(T)$ هي الاحتمالات الجزئية الموافقة لاحتمال تواجد المتحرك عند كل عقدة n من المخطاط بعد عدد معين من الخطى العشوائية T.

$$P(T)=\begin{pmatrix} P_{-E}(T) \\ \vdots \\ P_{-1}(T) \\ P_0(T) \\ P_1(T) \\ \vdots \\ P_E(T) \end{pmatrix} \qquad , \qquad \sum_{n=E}^{-E} P_n(T) = 1 \tag{2-04}$$

و بالتالي يمكننا في الأخير وصف ديناميكا المشي العشوائي عن طريق معادلة مصفوفية تربط بين نتائج احتمال الخطوة القديمة و نتائج احتمال الخطوة الجديدة عن طريق مصفوفة الانتقال M؛ أي:

$$\begin{gathered} P(T) = [c \cdot A]^T \cdot P(0) \\ P(T) = [M]^T \cdot P(0) \end{gathered} \tag{2-05}$$

حيث:

P: هو شعاع تَوَزُع الاحتمال الكلي. و يتكون من N مركبة الموافقة لـ N عقدة.

$P(0)$: هو شعاع توزع الاحتمال الكلي في اللحظة $t = 0$ أو بعد الخطوة $T = 0$.

$P(T)$: هو شعاع توزع الاحتمال الكلي بعد T خطوة معينة أو مختارة.

c: هو احتمال العُمْلَة. و الذي يأخذ قيمة سُلّمِيَة في حالة المشي الكلاسيكي، و تساوي قيمته (في حالة البنى المنتظمة) إلى مقلوب عدد الاتجاهات أو الخطوات الممكنة في المخطاط $\frac{1}{d}$.

A: هي مصفوفة الخطوات (أو الجوار كما يسميها البعض). و هي عبارة عن مصفوفة مربعة $(N\times N)$.

M: هي مصفوفة الانتقال (أو الاحتمال كما يسميها البعض) حيث أن مجموع عناصر كل عمود تساوي الوحدة (شرط انتظام المخطاط)، أي: $\sum_{i=-E}^{E} M_{ij} = 1$.

عمليا يمكننا القول أننا أنهينا عملية بناء النموذج و المعادلات الخاصة به، و في ما سيأتي سنستعرض بعض الأمثلة المهمة و المشهورة و نحلل نتائجها و نستخلص أهم مميزاتها.

2.2 البنى غير المنتهية كالخط المستمر (ذات الاتجاهين و الاتجاهات الثلاثة):

1.2.2 معادلات المشي العشوائي:

في حالة الخط المستمر (الوثيقة 2-01)، يمكن أن نعبر عن جميع الخطوات المتجهة نحو اليمين بالعبارة:

$$\left(\sum_{n=-E}^{E-1} |n+1\rangle\langle n|\right) + |-E\rangle\langle E| \tag{2-06}$$

و جميع الخطوات المتجهة نحو اليسار بالعبارة:

$$\left(\sum_{n=-E+1}^{E} |n-1\rangle\langle n|\right) + |E\rangle\langle -E| \tag{2-07}$$

و جميع خطوات الاستراحة بالعبارة:

$$\sum_{n=-E}^{E} |n\rangle\langle n| \tag{2-08}$$

و بالتالي تكون العبارة الشعاعية لمجموع خطوات كل من المخطاط (أ) و (ب) هما على الترتيب:

$$A=\left(\left(\sum_{n=-E}^{E-1} |n+1\rangle\langle n|\right)+|-E\rangle\langle E|\right)+\left(\left(\sum_{n=-E+1}^{E} |n-1\rangle\langle n|\right)+|E\rangle\langle -E|\right) \tag{2-09}$$

$$A=\left(\left(\sum_{n=-E}^{E-1} |n+1\rangle\langle n|\right)+|-E\rangle\langle E|\right)+\left(\left(\sum_{n=-E+1}^{E} |n-1\rangle\langle n|\right)+|E\rangle\langle -E|\right)+\left(\sum_{n=-E}^{E} |n\rangle\langle n|\right) \tag{2-10}$$

و بتطبيق الشكل التفصيلي لأشعة العقد المُعَرَّفَةِ مسبقا، يصبح مجموع الخطوات اليمنى و اليسرى الموافق للمخطاط (أ) في الوثيقة 2-01 يُعْطَى بمصفوفة مربعة صفرية باستثناء عناصر الصفين القطريين اللذين يقعان أعلى (توافق خطوات اليسار) و أسفل (توافق خطوات اليمين) الصف القطري للمصفوفة:

$$A = \begin{pmatrix} 0 & 1 & 0 & & 0 & 0 & 1 \\ 1 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 1 \\ 1 & 0 & 0 & & 0 & 1 & 0 \end{pmatrix} \tag{2-11}$$

و في حالة المخطاط (ب) من نفس الوثيقة، سَتُعْطَى مصفوفة مجموع الخطوات، بنفس المصفوفة المربعة السابقة للمخطاط (أ) زائد عناصر قطرية غير معدومة (توافق خطوات الاستراحة)، أي:

$$A=\begin{pmatrix}1&1&0&&0&0&1\\1&1&1&\dots&0&0&0\\0&1&1&&0&0&0\\&\vdots&&\ddots&&\vdots&\\0&0&0&&1&1&0\\0&0&0&\dots&1&1&1\\1&0&0&&0&1&1\end{pmatrix} \qquad (2\text{-}12)$$

أخيرا، الكتابة المصفوفية المفصلة لمعادلات المشي، الموافقة للمِخْطَاطَيْنْ (أ) و (ب) على الترتيب هما[2]:

$$P(T)=\begin{pmatrix}P_{-E}(T)\\\vdots\\P_{-1}(T)\\P_0(T)\\P_1(T)\\\vdots\\P_E(T)\end{pmatrix}=\left[\left(\frac{1}{2}\right)\cdot\begin{pmatrix}0&1&0&&0&0&1\\1&0&1&\dots&0&0&0\\0&1&0&&0&0&0\\&\vdots&&\ddots&&\vdots&\\0&0&0&&0&1&0\\0&0&0&\dots&1&0&1\\1&0&0&&0&1&0\end{pmatrix}\right]^T\cdot\begin{pmatrix}P_{-E}(0)\\\vdots\\P_{-1}(0)\\P_0(0)\\P_1(0)\\\vdots\\P_E(0)\end{pmatrix} \qquad (2\text{-}13)$$

$$P(T)=\begin{pmatrix}P_{-E}(T)\\\vdots\\P_{-1}(T)\\P_0(T)\\P_1(T)\\\vdots\\P_E(T)\end{pmatrix}=\left[\left(\frac{1}{3}\right)\cdot\begin{pmatrix}1&1&0&&0&0&1\\1&1&1&\dots&0&0&0\\0&1&1&&0&0&0\\&\vdots&&\ddots&&\vdots&\\0&0&0&&1&1&0\\0&0&0&\dots&1&1&1\\1&0&0&&0&1&1\end{pmatrix}\right]^T\cdot\begin{pmatrix}P_{-E}(0)\\\vdots\\P_{-1}(0)\\P_0(0)\\P_1(0)\\\vdots\\P_E(0)\end{pmatrix} \qquad (2\text{-}14)$$

2.2.2 الحساب اليدوي و كيفية تحرير البرنامج الحاسوبي:

يدويا يمكننا إجراء الحساب لعدد محدود من الخطوات الأولى فنجد مثلا في حالة المخطاط (أ) المبين في الوثيقة 2-01 النتائج:

العقد / الخطوات	$-E$	...	-4	-3	-2	-1	0	1	2	3	4	...	E
$P(0)$	0	...	0	0	0	0	**1**	0	0	0	0	...	0
$P(1)$	0	...	0	0	0	**1/2**	0	**1/2**	0	0	0	...	0
$P(2)$	0	...	0	0	**1/4**	0	**2/4**	0	**1/4**	0	0	...	0
$P(3)$	0	...	0	**1/8**	0	**3/8**	0	**3/8**	0	**1/8**	0	...	0
$P(4)$	0	...	**1/16**	0	**4/16**	0	**6/16**	0	**4/16**	0	**1/16**	...	0

أما في حالة المخطاط (ب) ذو الثلاثة اتجاهات فسنجد النتائج:

[2] هذا الشكل أو الهيئة المصفوفية المفصلة تعمدت إظهارها لتفيدنا في فهم آلية حدوث الحسابات و بالتالي كيفية تصميم البرنامج الحاسوبي.

العقد / الخطوات	$-E$	⋯	-4	-3	-2	-1	0	1	2	3	4	⋯	E
$P(0)$	0	⋯	0	0	0	0	**1**	0	0	0	0	⋯	0
$P(1)$	0	⋯	0	0	0	**1/3**	**1/3**	**1/3**	0	0	0	⋯	0
$P(2)$	0	⋯	0	0	**1/9**	**2/9**	**3/9**	**2/9**	**1/9**	0	0	⋯	0
$P(3)$	0	⋯	0	**1/27**	**3/27**	**6/27**	**7/27**	**6/27**	**3/27**	**1/27**	0	⋯	0
$P(4)$	0	⋯	**1/81**	**4/81**	**10/81**	**16/81**	**19/81**	**16/81**	**10/81**	**4/81**	**1/81**	⋯	0

أما في حالة عدد خطوات كبير نسبيا يمكن أن نستعين ببرنامج حاسوبي للقيام بذلك (الملحق: البرنامج 1). و قد استعملت هنا لغة البرمجة البايثون (Python)، و هي لغة برمجة مجانية و سهلة و واعدة، كما ان أوامرها تشبه كثيرا العبارات الرياضياتية التي نكتبها. و لكن قبل استعراض نتائج البرنامج و تحليلها، هناك مجموعة من التنبيهات المهمة التي يجب الإشارة إليها، لضبط و فهم كيفية بناء و استعمال البرنامج.

<u>تنبيهات هامة:</u>

1) عادة ما يكون ترقيم العقد من 0 إلى N-1، و ذلك لسهولة التعامل معها أثناء الحساب و البرمجة. و عليه سنغير ترقيم العقد في الوثيقة 01-2، من $-E$, ..., 0, ..., E. إلى 0, ..., $\frac{N-1}{2}$, ..., N-1 على الترتيب أثناء تحرير البرنامج. حيث سيبقى العدد الكلي للعقد هو نفسه ($N=(2E+1)$). و بداية الحركة ستكون من العقدة المرقمة بـ $\frac{N-1}{2}$ في الترقيم الجديد عوض العقدة 0 في الترقيم القديم.
2) تعريفيا، البنى المنتظمة هي كلها بنى منتهية (مغلقة). و لكن تقنيا، يمكننا محاكات خصائص البنى غير المنتهية (المستمرة) عن طريق البنى المنتظمة المناسبة لها (فمثلا الحلقة المغلقة؛ و التي تعتبر بنية منتظمة، تناسب الخط المفتوح). و لكن مع الحرص على تفادي حدوث التداخل بين مختلف جهات انتشار الاحتمال الممكنة. و ذلك بجعل عدد العقد يتعلق دائما بتغير عدد الخطوات المختار، أي: $N \geq f(T)$ (شكل الدالة $f(T)$ تحدده خصائص البنى المدروسة). فمثلا في حالة الخط المفتوح، و بعد تثبيتنا لعدد الخطوات المختار T. يجب أن يكون عدد العقد أكبر من ضعف عدد الخطوات (لأن الخطوات تنتشر في اتجاهين) زائد 1 (بسبب العقد الصفرية). أي: $N \geq (2T+1)$. و بذلك يمكننا أن نتفادى دوما (و مهما كانت قيمة الخطوات T التي سنختارها) حدوث التداخل بين انتشار الاحتمال الذي ينتشر جهة اليمين مع الاحتمال الذي ينتشر جهة اليسار بسبب انغلاق البنية. و

نكون في الاخير قد حاكينا سلوك بنية غير منتهية (مثل خط مفتوح) عن طريق بنية منتظمة و منتهية (مثل الحلقة).

3) فيما يخص هذا البرنامج يمكن نسخه ثم لصقه على أي صفحة جديدة لقارئ لغة البايثون و منفذها و سيعمل مباشرة.

4) في بعض الحالات يمكن أن يتم اللصق بشكل غير صحيح (هذا ما حدث معي في بعض الحالات). فلذا أنصح بمتابعة جميع السطور و تصحيح الأشياء التي لم تلصق بالشكل الصحيح. لأن البرنامج صحيح و يعمل معي دون مشاكل أو أعطاب.

5) يجب تثبيت جميع الحزم المطلوبة على الحاسب مثل الحزمة "numpy" و "matplotlib.pyplot" إذا كانت غير مثبة.

6) البرامج التي تقرأ لغة البايثون و تنفذها كثيرة. و أنا استعملت هنا القارئ Enthought canopy وهو مجاني و خاصة للاستعمالات الأكاديمية حيث يوفر خدمة ما يسمى بـ free academic uses

7) الكتابات التي تأتي بعد العلامة "#" أو بينها هي مجرد تعليقات في لغة البايثون. إذن فهو لا يقوم بتنفيذها.

3.2.2 تحليل المنحنيات و استخلاص النتائج:

بعد تنفيذ البرنامج سنجد أن النتائج (الوثيقة 02-2) توضح جليا أن جل الاحتمال سيبقى بجوار الموضع الأصلي الذي انطلق منه المشي العشوائي (العقدة 0)، و لا يبتعد كثيرا عنه مقارنة بعدد الخطوات التي مشاها. حيث أن:

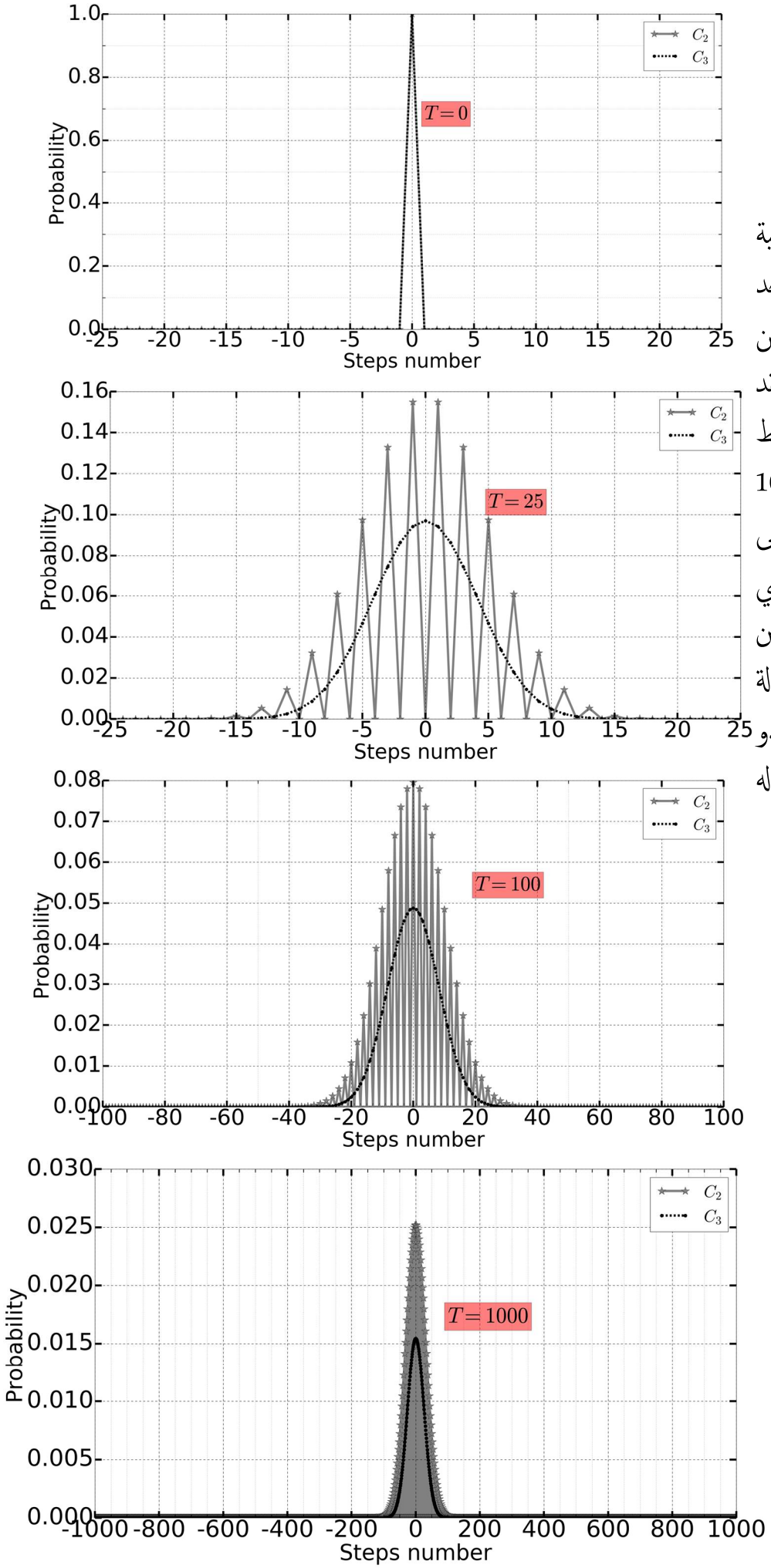
1) قِمَّة الاحتمال ستكون متمركزة دوما في عقدة الانطلاق في حالة المخطاط (ب)، و تتمركز في عقدة الانطلاق أو العقدتان اللتين تليانها مباشرة في حالة المخطاط (أ). و ذلك حسب طبيعة عدد الخطوات، زوجية كانت أم فردية على الترتيب.

2) في حالة المخطاط (أ) تنعدم قيمة الاحتمال في العقد الزوجية إذا كان عدد الخطوات فردي و العكس صحيح. أما في حالة المخطاط (ب) فلا تنعدم قيمة الاحتمال مهما كان نوع عدد الخطوات.

3) عموما توزيع الاحتمال خاصة في حالة عدد كبير من الخطى يتقارب إلى عبارة توزيع غوص.

الوثيقة 02-2:

التمثيل البياني لكيفية تطور احتمال تواجد المتحرك (بعد انطلاقه من العقدة 0) عبر مختلف عقد المخطاط على شكل خط مفتوح بعد 0, 25, 100 و 1000 خطوة على الترتيب. في حالة المشي الكلاسيكي ذو الاتجاهين (نرمز له بـ: C_2)، في حالة المشي الكلاسيكي ذو الثلاث اتجاهات (نرمز له بـ: C_3).



مع فرق بسيط (عبارة توزيع غوص مضروب في 2). أي:

$$P_n(T) \cong 2 \cdot \left(\frac{1}{\sqrt{2\,\pi\,\sigma^2}} \cdot e^{\frac{(n-\mu)^2}{2\,\sigma^2}} \right) \quad , \quad \begin{matrix} \sigma^2(T) = T \\ \mu(T) = 0 \end{matrix} \qquad (2\text{-}15)$$

3) تتناقص قيمة قِمَّةُ الاحتمال مع تزايد عدد الخطوات، في حين يزداد عرض بيان الاحتمال الكلي.

4) كَمِيًّا و في هذا النوع من البنى المفتوحة، يمكن أن نقيس أو نقدر تشتت مركبات الاحتمال الكلي عن طريق التباين أو الانحراف المعياري، حيث نجد أنه يتناسب بالتقريب مع عدد الخطوات أو جذره التربيعي على الترتيب [8] أي:

$$\sigma^2(T) = T \Rightarrow \sigma(T) = \sqrt{T} \qquad (1\text{-}16)$$

5) بمقارنة ما مدى تشتت نتائج المخطاط (أ) مع نتائج المخطاط (ب)، يمكننا أن نقول أن المخطاط (أ) هو الأكثر تشتتا، أو هو الأسرع ابتعادا، عن موضع انطلاقه بدلالة عدد الخطوات، مقارنة مع المخطاط (ب).

3.2 البنى المنتهية كالحلقة (ذات الاتجاهين و الاتجاهات الثلاثة):

1.3.2 معادلات المشي العشوائي:

في حالة البنى المغلقة كالحلقة مثلا. سنستخدم نفس الوثيقة 2-01 و بالتالي نفس معادلات البنى غير المنتهية السابقة. حيث سنستخدم المعادلة 2-13 في حالة المشي وفق اتجاهين و المعادلة 2-14 في حالة المشي وفق ثلاث اتجاهات، و لكن مع الحرص على حدوث ظاهرة التداخل هذه المرة، بين جهات انتشار الاحتمال المختلفة، و ذلك بتحرير عدد العقد الكلي N من تبعيته لعدد الخطوات الكلي T، و جعله ثابتا في قيمة معينة ما بحسب الاختيار.

2.3.2 الحساب اليدوي و كيفية تحرير البرنامج الحاسوبي:

في الحساب اليدو، سنجد أن سلوك الاحتمال الكلي في حالة الحلقة هو نفسه سلوكه في حالة الخط، و لكن هذا فقط بالنسبة للخطوات الأولى قبل أن يصل انتشار الاحتمال إلى العقدتين النهائيتين E و $E-$، و بعدها سيتغير سلوك الاحتمال جراء بداية التداخل بين انتشار احتمال جهة اليمين مع انتشار احتمال جهة اليسار. فمثلا عندما نثبت E عند القيمة $E = 3$ نبدأ نلاحظ الفرق انطلاقا من الخطوة $T = 4$ أين يبدأ التداخل بين انتشار احتمال اليمين و اليسار سواء في حالة المخطاط (أ):

العقد / الخطوات	-3	3	2	1	0	-1	-2	-3	3
$P(0)$	0	0	0	0	**1**	0	0	0	0
$P(1)$	0	0	0	**1/2**	0	**1/2**	0	0	0
$P(2)$	0	0	**1/4**	0	**2/4**	0	**1/4**	0	0
$P(3)$	**1/8**	**1/8**	0	**3/8**	0	**3/8**	0	**1/8**	**1/8**
$P(4)$	**1/16**	**1/16**	**4/16**	0	**69/16**	0	**4/16**	**1/16**	**1/16**

أو في حالة المخطاط (ب):

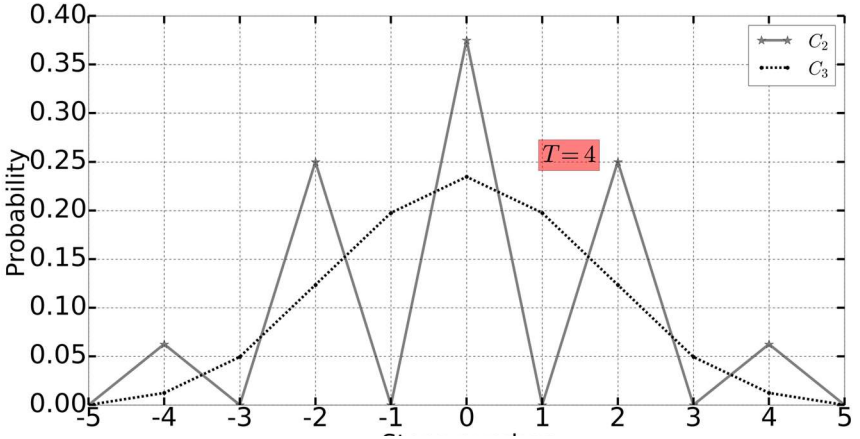
العقد / الخطوات	3	-3	-2	-1	0	1	2	3	-3
$P(0)$	0	0	0	0	**1**	0	0	0	0
$P(1)$	0	0	0	**1/3**	**1/3**	**1/3**	0	0	0
$P(2)$	0	0	**1/9**	**2/9**	**3/9**	**2/9**	**1/9**	0	0
$P(3)$	**1/27**	**1/27**	**3/27**	**6/27**	**7/27**	**6/27**	**3/27**	**1/27**	**1/27**
$P(4)$	**5/81**	**5/81**	**10/81**	**16/81**	**19/81**	**16/81**	**10/81**	**5/81**	**5/81**

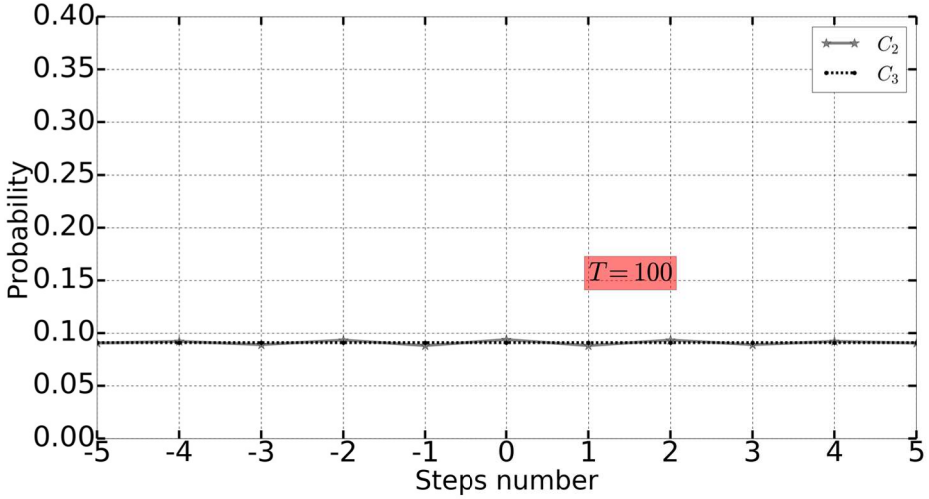
أما في حالة عدد خطوات كبير نسبيا، سوف نحافظ إجمالا على نفس البرنامج السابق للخط المفتوح (الملحق: البرنامج 1)، و لكن مع تغير واحد مهم و هو تحرر عدد العقد من تبعيته لعدد الخطوات و التي كانت وفق العلاقة: $N \geq (2T+1)$ و نثبته في قيمة معينة مختارة ثابتة (و لتكن مثلا N=11 عقدة) و نرى كيف يتصرف سلوك الاحتمال الكلي قبل و بعد حدوث التداخل.

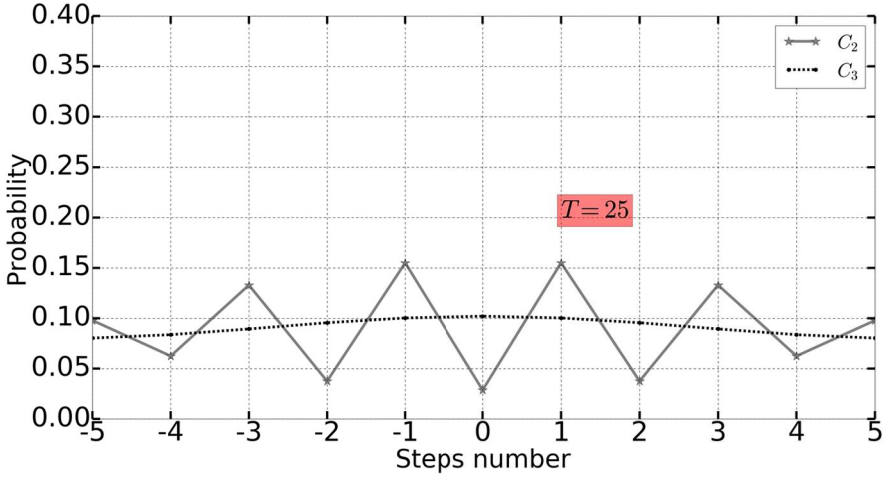
3.3.2 تحليل المنحنيات و استخلاص النتائج:

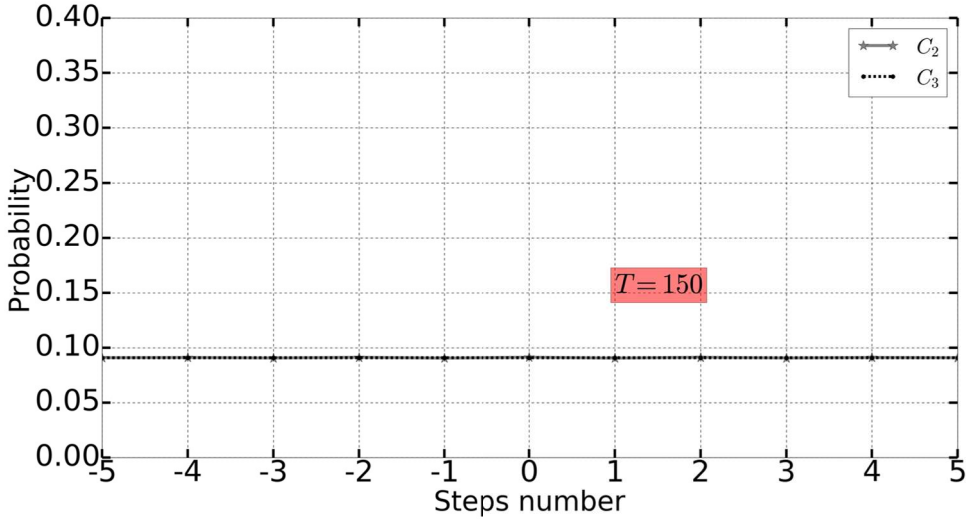
بعد تنفيذ البرنامج سنجد أن النتائج (الوثيقة 03-2) توضح جليا أن: (1) سلوك الاحتمال قبل وصوله إلى العقد الطَرَفِيَةِ سيكون تماما مثل سلوك الخط المفتوح. (2) و لكن بعد تداخل الاحتمال الذي ينتشر من جهة اليمين مع الاحتمال الذي ينتشر من جهة اليسار، نلاحظ أن الفرق في السلوك بين الحلقة و الخط المستمر بدأ يظهر، حيث أن الاحتمال يبدأ تدريجيا بالميل إلى التوزع بشكل منتظم و متساو على جميع العقد و هذا ما يحصل بالفعل. (3) و بعدها سيبقى هذا الانتظام أو الاستقراء على هذه الحال مهما زدنا عدد خطوات أو مدة المشي. (4) و لكن من جهة أخرى، نجد أن الزمن أو عدد الخطوات اللازم لحدوث هذا الاستقرار سوف يختلف من مخطاط لآخر، حيث نجد أن المخطاط (ب) سيكون أسرع في وصوله إلى حالة الاستقرار مقارنة بالمخطاط (أ).

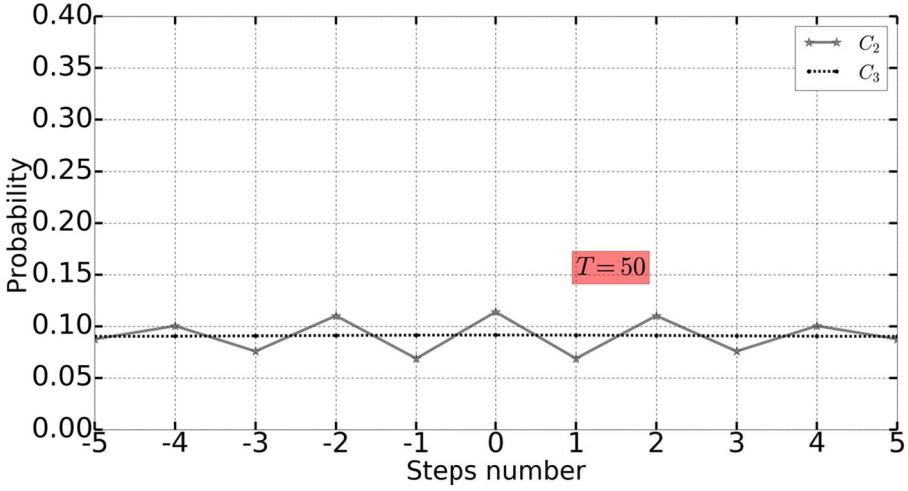
الوثيقة 03-2: التمثيل البياني لكيفية تطور احتمال تواجد المتحرك (بعد انطلاقه من العقدة 0) عبر مختلف العقد لمخطاط على شكل حلقة مغلقة تحتوي على 11 عقدة بعد 4، 25، 50، 100 و 150 خطوة على الترتيب. في حالة المشي الكلاسيكي ذو الاتجاهين (نرمز له بـ: C_2)، في حالة المشي الكلاسيكي ذو الثلاث اتجاهات (نرمز له بـ: C_3).











3.2 المقارنة بين مختلف البنى و المخاطيط المنتهية (فكرة المستويات):

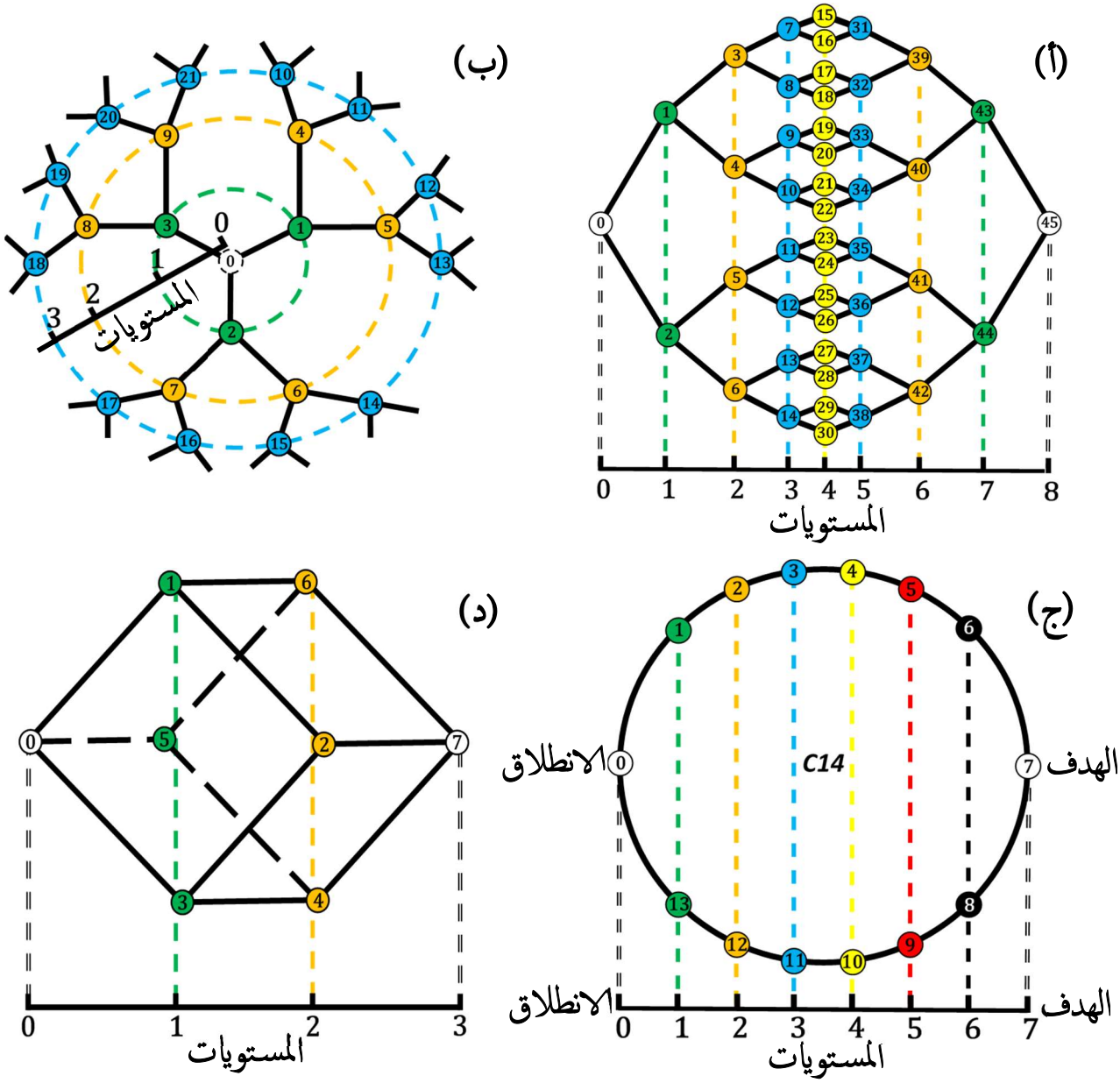
كما رأينا سابقا، فآلية الاختبار التي استعملت في الخط المفتوح هي غير آلية الاختبار التي استعملت في الحلقة المغلقة. و لهذا لا يمكننا المقارنة بينهما بسبب اختلاف طبيعة هذه البُنى و سلوك نتائجها. و لكن رغم هذا، فعمليا هناك طريقة أخرى تمكننا من تكييف هذه البنى المختلفة إلى نمط بنيوي معين، يكون مشترك بينها جميعا، و بالتالي يمكننا المقارنة بينها عن طريق مختلف وسائل الاختبار الممكنة. يعتبر نمط المستويات حلا في مثل هذه الحالات، حيث كل بنية مهما كانت هيئتها يمكن تكييفها إلى بنية أخرى تكافئها. فعوض أن نتعامل مع العقد مباشرة سنتعامل مع ما نسميه بالمستويات، حيث كل مستوى في المخطاط الجديد سيكون مكافئاً لسلوك عقدة واحدة أو مجموعة من العقد من المخطاط الأصلي. إذْ يمكن أن نُعَرِفَ كل مستوى على أنه كل العقد الجديدة التي يمكن أن يصلها المتحرك لأول مرة بعد كل خطوة جديدة إلى الأمام مبتعدا عن نقطة انطلاقه (الوثيقة 04-2). بعد تحديد المخطاط المكافئ لكل البنى المختلفة، يمكننا حينها المقارنة بين مستوياتها. هنا سوف ندرس فقط حالة الحلقة و الخط المغلق، حيث ستكون الحلقة ممثلة بخط مغلق يصف مستوياتها (الوثيقة ج-04-2) و الخط المغلق العادي سيكون ممثل بعقده العادية (كل عقدة تعتبر مستوى).

4.2 خاصية الانتقال (المقارنة بين الخط المنتهي و الحلقة):

للمقارنة بين الحلقة و الخط المحدود هناك عدة خصائص ممكنة للمشي العشوائي يمكننا أن نختبرها. و لقد جربنا عدة خصائص أو مقادير (راجع الفصل الأول) و لكن وجدنا أن خاصية الانتقال هي التي تعطي أوضح و أفضل النتائج بالنسبة لموضوع الأطروحة ككل. و لذا سوف نقتصر في دراستنا هنا على اختبار خاصية الانتقال و حسب.

1.4.2 تعريف خاصية الانتقال و شرح كيفية حسابها:

هي كمية نحسب من خلالها سرعة انتقال الاحتمال بين عقدة أو مستوى معين يتم اختياره كموضع انطلاق (لنقل العقدة 0) ينطلق منه الاحتمال الكلي و بين عقدة أخرى أو مستوى آخر يتم اختياره كموضع وصول أو هدف (لنقل العقدة a، عادة تختار في نهاية الطرف المقابل لموضع الانطلاق)، بحيث

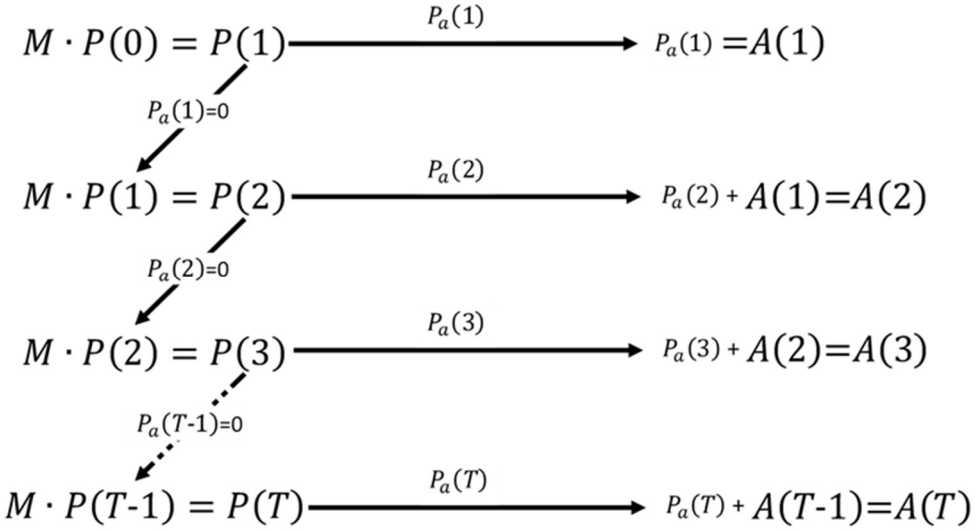


الوثيقة 04-2: هنا لدينا مجموعة من البنى أو المخاطيط ذات العقد و مُكَافِئاتُهَا على شكل خط منتهٍ ذي مستويات: **(أ)** مخطاط على شكل تفرع شجيري "ثنائي، متقابل الوجوه". الوجه الأول من العقدة 0 إلى غاية العقدة 30 متداخل مع الوجه الثاني من العقدة 15 إلى العقدة 45. يمكن مكافئة هذا المخطاط بخط محدود ذي تسع مستويات **(ب)** مخطاط نجمي: هنا لدينا حالة النجمة الثلاثية، حيث كل عقدة لها ثلاث وصلات. يمكن مكافئة هذا المخطاط بخط نصف محدود ذي أربع مستويات. و إذا جمعنا هذا المخطاط على أساس أنه الوجه الأول مع وجهه الثاني (الذي ينعكس في المرآة) نتحصل على بنية مغلقة على شكل كرة بنيتها المكافئة ستكون على شكل خط محدود بسبع مستويات. **(ج)** مخطاط دائري أو حلقي يتكون من عدد زوجي من العقد يمكن مكافئته بخط محدود ذي ثماني مستويات. **(د)** مخطاط على شكل مكعب بسيط يمكن مكافئته عن طريق خط محدود ذي أربع مستويات. و يمكن تعميم هذه الفكرة لأي مخطاط أو بُنَى مهما كان تعقيدها أو نوعها.

بعد بداية المشي العشوائي، سينتشر الاحتمال عبر العقد و المستويات شيئا فشيئا إلى غاية أن يبدأ في الوصول إلى الهدف، الذي سنزوده بخاصية الامتصاص، حيث كل قِيمَةِ احتمال تصل إليه، فلا يمكنها أن تغادره مرة أخرى [3] إذن هكذا و دواليك إلى غاية أن يَحْتَجِزَ الهدف كل الاحتمال الكلي. رياضياتيا يمكن أن نعبر عن هذه الفكرة بدالة تَنْتَزِعُ قيمة الاحتمال التي تصل إلى العقدة a بعد كل خطوة، ثم تجميعها في مقدار جديد هو:

$$A(T) = \sum_{t=0}^{T} P_a(t) \qquad (2\text{-}17)$$

الوثيقة 05-2 توضح فكرة حساب خاصية الانتقال في شكل مخطط توضيحي خطوة بعد خطوة. و بالتالي يمكننا في النهاية و بعد حساب تطور الدالة $A(T)$ لكل بنية من معرفة من هو المخطاط الأسرع في انتقال (امتصاص أو احتجاز) كل الاحتمال الكلي المنطلق من موضع الانطلاق.



الوثيقة 05-2: مخطط توضيحي، يظهر عن طريق معادلات المشي الكلاسيكي، كيفية حساب و تطور الاحتمال الممتص $A(T)$ عند موضع الهدف (العقدة a) انطلاقا من الخطوة 0 ثم خطوة بخطوة إلى غاية الخطوة T.

[3] لتوضيح الفكرة سنمثل مكان الهدف هنا كحفرة عميقة كلما وصلت إليها أي قيمة من الاحتمال الكلي المنتشر فستسقط فيها و لن تخرج منها أبدا.

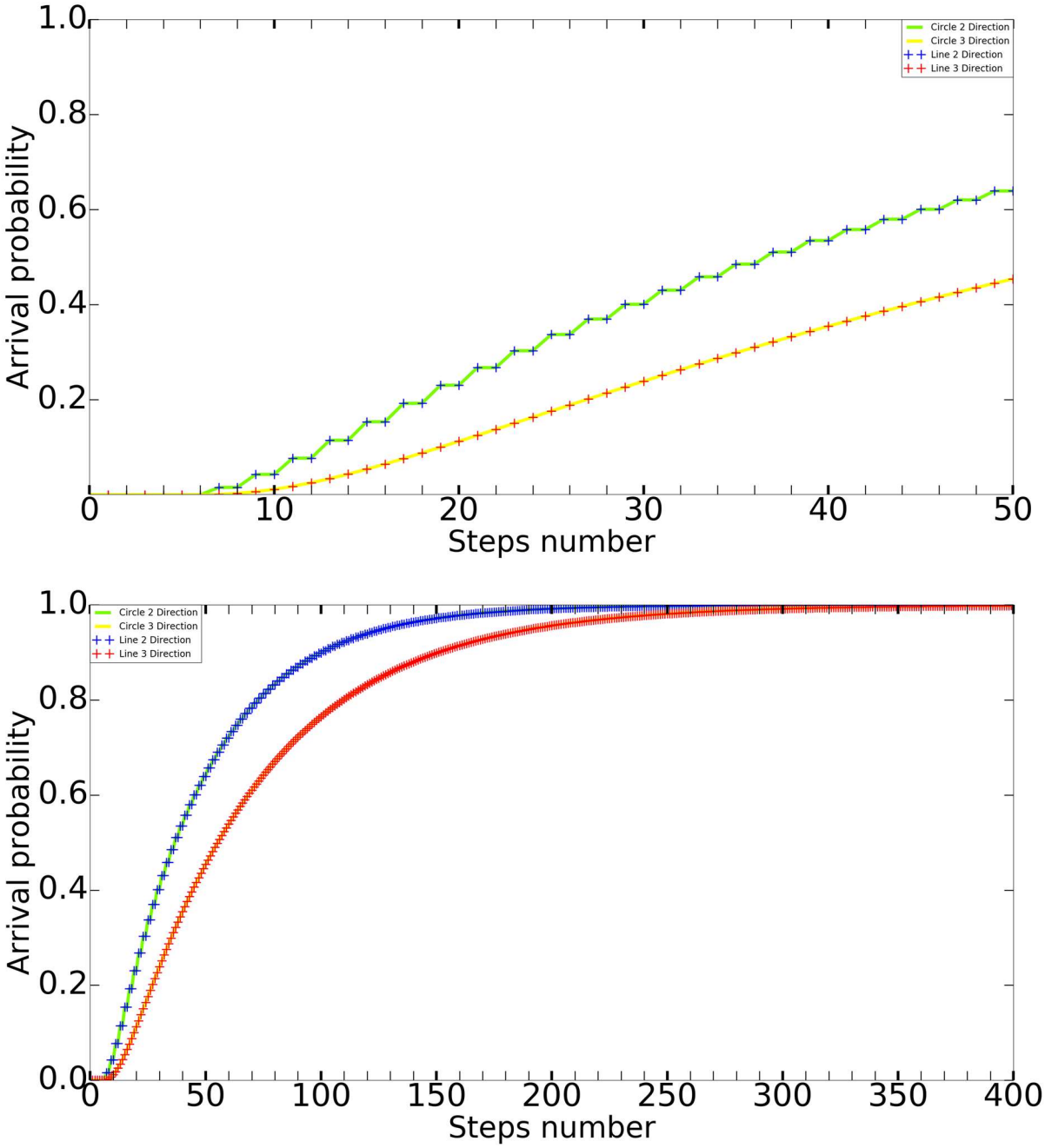
2.4.2 كيفية تحرير البرنامج الحاسوبي:

فكرة تحرير البرنامج ستكون بسيطة، حيث سنحافظ على نفس خطة بناء البرامج السابقة لآلية حساب تطور الاحتمال الكلي لكل من الحلقة و الخيط المحدود مع بعض التغييرات و الملاحظات التي سنلخصها في ما يلي:

1. سنتعامل هنا مع المخطاط الموضح في الوثيقة ج-04-2 عوض الوثيقة 01-2. حيث سنقارن بين الحلقة ذات 14 عقدة (المكافئ الخط المغلق ذو الـ 8 مستويات) و الخط المحدود ذو الـ 8 عقد.
2. للحصول على خط محدود من جهة اليسار (هذه الفكرة يمكن تعميمها على أي جهة نريد غلقها) سنضع عارضة انعكاس بالنسبة للخط المفتوح عند موضع انطلاقه في العقدة 0. حيث لا يمكن لاحتمال أن ينتشر على يسار العقدة 0. إذْ سنعمد لتحويل الخطوة التي ستنطلق من العقدة 0 متجهة يسارا إلى الاتجاه يمينا. و بالتالي العقدة 0 سيكون لديها خطوتين في جهة اليمين عوض خطوة واحدة يمينا و واحدة يسارا.
3. في ما يخص الحلقة فخطها المكافئ هو مغلق أصلا، لأن الاحتمال الذي يكون في المستوى 0 (المكافئ للعقدة 0) سيَصُبُ حتما كله في المستوى 1 (المكافئ للعقدتين اللتين تأتيان قبل و بعد العقدة 0، أي العقدة 1 و العقدة 13 على الترتيب).
4. سنضع عارضة امتصاص عند موضع الهدف (العقدة a)، لكل من الخط المحدود عند العقدة 7 و الحلقة المغلقة عند المستوى رقم 7 المكافئ للعقدة 7 (الوثيقة ج-04-2) و ذلك بانتزاع أي قِيمَةْ احتمال تصل إلى العقدة الهدف بعد نتيجة توزع احتمال كل خطوة (الانتزاع سيتم على مستوى مركبة شعاع الاحتمال الكلي الموافقة للعقدة 7 بنسبة للخط المحدود و الحلقة سواء).
5. خطة الحساب ستتم بالضبط وفق ما هو موضح في الوثيقة 05-2.

3.4.2 تحليل المنحنيات و استخلاص النتائج:

بعد تنفيذ البرنامج (الملحق: البرنامج 2) و الحصول على النتائج (الوثيقة 06-2) لكل من الخط المحدود ذي الثماني عقد و الحلقة ذات 14 عقدة سواء في حالة الاتجاهين أو الثلاث اتجاهات، سنجد أن (1) سرعة امتصاص الاحتمال الكلي هي نفسها في الخط المحدود و الحلقة. (2) و لكن نجد أن حالة الثلاث اتجاهات تكون أبطئ من حالة الاتجاهين في سرعة انتقال او امتصاص الاحتمال الكلي.



الوثيقة 06-2: تطور قيمة الاحتمال الممتص (عند العقدة الهدف) في حالة الحلقة و الخط المغلق ذَوَا الثمانية مستويات (الوثيقة ج-04-2) بعد 50 ثم 400 خطوة، حيث نلاحظ أن الحلقة مكافئة تماما للخط المغلق من حيث سرعة انتقال الاحتمال، و لكن من جهة أخرى، يتضح أن انتقال الاحتمال يكون في حالة الاتجاهين أسرع من أن نظيف خطوة استراحة كاتجاه ثالث.

الفصل الثالث:

المشي الكمي

في هذا الفصل سنستعرض فكرة المشي العشوائي الكمي و كيفية استخلاصه انطلاقا من فكرة المشي الكلاسيكي، و التي تخالف نوعا ما الطريقة المألوفة في تكميم الأنظمة الكلاسيكية لتصبح أنظمة كمية [4]: حيث عوض أن نتحدث عن تعويض كميات كلاسيكية مثل العزم أو الطاقة بمؤثرات كمية واحدية في فضاء هيلبرت سنعوض احتمال العملة الكلاسيكي الذي كان عبارة عن قيمة سلمية و فقط بمؤثر كمي واحدي جديد، سميّ "بمؤثر العملة" يؤثر في فضاء هيلبرت، أشعة وِحْدَتِهِ التي ستعبر عن مختلف الوصلات التي تنطلق من كل عقدة، سيكون لها الدور العميق في مساعدتنا على بناء المعادلات الزمنية لتطور المشي العشوائي الذي يستوعب أحد أهم مبادئ ميكانيك الكم كمبدأ التراكب (التواجد في عدة أماكن في نفس اللحظة).

هذا الفصل يمكن اعتباره مجازا، النسخة المكممة للفصل الأول، حيث نفس تسلسل كل ما درس كلاسيكيا في الفصل الأول، سيتم بنفس التسلسل دراسته كميّا في هذا الفصل. و بالتأكيد التركيز و التنبيه و التعليق على الفروقات الجوهرية بين النسختين، بحسب الضرورة و الحاجة. و عدم تكرار شرح و توضيح ما شرح و وضح في الفصل الأول.

في ما يخص المراجع يعتبر المرجع [1] من أشهر الأعمال التي أعطت نظرة سريعة و شاملة و وافية لحدٍ ما، لفهم فكرة المشي الكمي و تطبقاته. ثم يأتي المرجع [8] الذي يمكن الرجوع إليه لتفصيل و التعمق أكثر في البرهنة لأهم النتائج و التفاصيل. و أخيرا فيما تعلق بالتطبيقات و استعمالات المشي الكمي المرجع [9] يعتبر وافيا و كافيا لمن أراد التعمق.

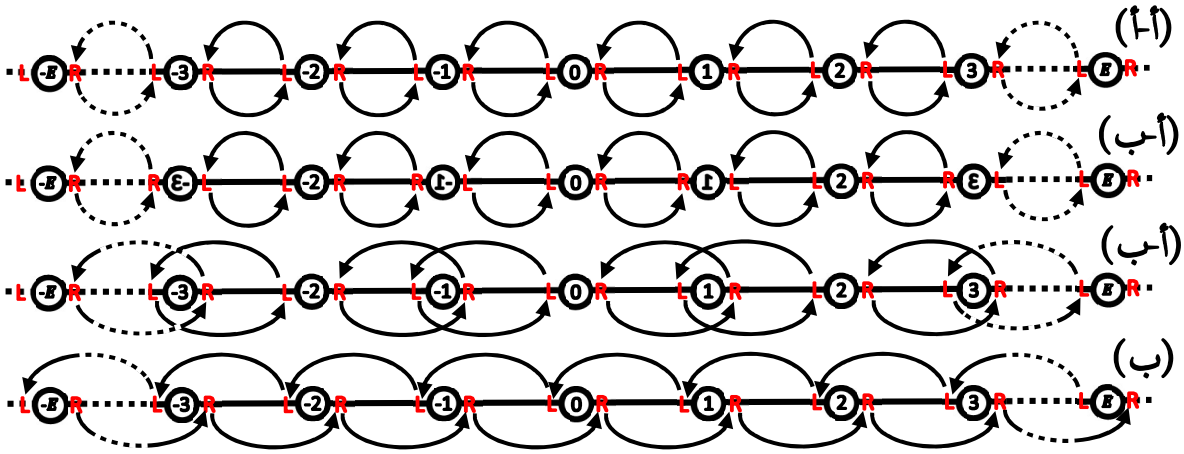
---

[4] ربما لأن المشي العشوائي غير المستمر في حد ذاته مكمم في الفضاء. أي أن الخطوات تتم بشكل غير مستمر و مكمم أصلا. و لذا كانت الخطة ربما هي تصميم معادلة تستوعب السلوك التطوري لميكانيك الكم. من جهة أخرى يمكن التنبيه الى أن طريقة إيجاد النسخة الكمية للمشي العشوائي المستمر تختلف عن طريقة إيجاد النسخة الكمية للمشي العشوائي غير المستمر [1، 8].

1.3 بناء النموذج في الحالة العامة (البنى المنتظمة):

في الحقيقة للحصول على المشي العشوائي الكمي الذي يعتبر النسخة المكممة و المقابلة لمشي العشوائي الكلاسيكي، سنطبق نفس الفكرة الأساسية و التي تتمثل في عشوائية حركة المشي. و لكن هذه المرة، سيحكم تطورها الزمني و مبادئ ميكانيك الكم، عوض مبادئ الميكانيك الكلاسيكية. و منه لبناء النموذج الكمي للمشي العشوائي سنحافظ عموماً على مجمل التعاريف التي بنينا بها النموذج الكلاسيكي، و لكن هذه المرة، لن نكتفي بوصف المخطاط عن طريق مختلف عقده فقط. بل سنحتاج لإدراج فضاء شعاعي جديد، مستقل عن فضاء العقد. نعبر به عن عدد و تفاصيل مختلف أَطْرَافِ الوَصْلاَتِ التي تنطلق من كل عقدة على حدى. أَشِعَةُ وِحْدَةِ هذا الفضاء الشعاعي الجديد، ستساعدنا على تكييف المعادلات الزمنية لمشي العشوائي عند كل عقدة لتتأقلم وتستوعب ديناميكا ميكانيك الكم. إذن فالفضاء الشعاعي الكلي، الذي سيصف بنية أي مخطاط بكل عُقَدِهِ و تفاصيل أَطْرَافِ وَصْلاَتِهَا (الوثيقة 3-01) التي تتم عبرها الخطوات هو:

$$H = (H_N \otimes H_d) \qquad (3\text{-}01)$$



الوثيقة 3-01: مخطاط ذو اتجاهين لكل عقدة (العقدتين E و $-E$ متصلتين ببعضهما البعض للحصول على مخطاط منتظم). نُظهر فيه مجموعة من كيفية توزيع ترقيم العقد و أطراف وصلاتها وفق ثلاث مناهج مختلفة: **(أ)** على شكل ثنائيات ذهاب و إياب بين كل عقدتين متجاورتين: **(أ-أ)** مع طرفين مختلفين يمينا-يسارا "R-L" دائمًا. **(أ-ب)** مع طرفين متماثلين يمينا-يمينا "R-R" ثم يسارا-يسارا "L-L" حسب رتبة عقدة الانطلاق زوجية ثم فردية على الترتيب. **(ب)** على شكل مستمر و منفصل، حيث الذهاب في اتجاه اليمين مع أطراف يمينية "R-R" دائمًا و الإياب في اتجاه اليسار مع اطراف يسارية "L-L" دائمًا.

H_N: هو الفضاء الجزئي الذي يعبر عن عقد المخطاط، و بالتالي فإن بعده و عدد أشعة وحدته ($|n\rangle$، $n \in \mathbb{Z}_N$) يحدد وفق عدد هذه العقد "N".

H_d: هو الفضاء الجزئي [5]الذي سيعبر عن أطراف الوصلات التي تنطلق من كل عقدة، و بالتالي سيتم تحديد بُعْدِهِ و عدد أشعة وحدته ($|c\rangle$، $c \in \mathbb{Z}_d$) عن طريق عدد الوصلات "d" التي تنطلق من كل عقدة على حدى (في حالة الوثيقة 01-3 كل عقدة لها طرفين مما يعني أن: d=2).

عادة ما تُرَقَّمُ العقد و أشعة الوحدة الموافقة لها من 0 إلى غاية (N-1). و ترقم أطراف وصلاتها و أشعة الوحدة الموافقة لها من 0 إلى غاية (d-1) ؛[6] و تكون عملية توزيعها عبر المخطاط كيفية و اختيارية. و لكن لتسهيل عملية بناء معادلات الحركة و التعامل معها، أثناء دراسة البنى ذات الحجم الكبير خاصةً، يُعْمَدُ إلى اختيار توزيعها وفق نمط مُمَنْهَجٍ و مُقَعَّدٍ ما، غالبا ما يتم تعريفه في شكل ثنائيات، تكون بين كل خُطْوَةِ ذَهَابٍ و إِيَابٍ، بين كل عقدتين متواصلتين بوصلة (الوثيقة 01-3 توضح فكرة هذا التوزيع في حالة d=2). رياضياتيا يتم التعبير عن هذا التوزيع عن طريق تأثير دالة معينة (نسميها "e") تُوَزِّعُ الترقيم في شكل ثنائيات ذهاب و إياب. فمثلا كل عقدة n من جهة طَرَفِ وَصْلَتِهَا a تتصل بالعقدة v من جهة طَرَفِ وَصْلَتِهَا b عن طريق خطوة الذهاب، و العكس صحيح بالنسبة لخطوة الإياب، أي:

$$e(a,n) = (b,v) \qquad (3\text{-}02\text{-أ})$$

$$e\big(e(a,n)\big) = e(b,v) = (a,n) \qquad (3\text{-}02\text{-ب})$$

المشي الكمي يمكن تصوره و دراسته إذن، عبر حركة أي جسيمة ما، وفق خطوات عشوائية متماثلة. تتم خلال مجالات زمنية متساوية t. عبر مجموعة من المواضع أو الحالات المحددة N (أي: فضاء العقد). كل موضع أو حالة يحتمل درجة حرية داخلية أو ذاتية قيمتها d (أي: فضاء العملة) ؛[7] و عليه، فدالة

---

[5] سنسميه فضاء العُمْلَة، نسبة لمؤثر جديد سيتخذ أشعة وحدة هذا الفضاء كأشعة ذاتية له. هذا المؤثر الجديد سيلعب نفس دور العملة النقدية في الحالة الكلاسيكية. و بالتالي اشتقت تسميته من دوره و سمي بـ: مؤثر "العُمْلَة".

[6] أثناء الشرح اتبعت الترقيم مع العقد و الترميز الأَحْرُفِ مع أطراف الوصلات. و ذلك لسهولة فَهْمِ و تخيل الأفكار بشكل أفضل. و لكن في البرامج الحاسوبية اتبعت الترقيم الذي يُعْمَلُ به غالبا لأنه يسهل التحكم أكثر في البرمجة.

[7] في حالة البعدين "d=2" يمكن أن نعبر عن فضاء العملة و حالتيه لتقريب المعنى: بفضاء السبين و حالتيه (علوي، سفلي) أو بفضاء استقطاب الفوتونات و حالتيها (عمودي، أفقي) ...إلخ.

الحالة المقننة $\psi(t)$ التي ستصف تطور حالة جسيمتنا في فضاء هيلبرت الكلي H هي:

$$|\psi(t)\rangle = \sum_{n,c=0}^{N\text{-}1,d\text{-}1} \alpha_{n,c}(t)\,(|c\rangle\otimes|n\rangle) \quad , \quad \sum_{n,c=0}^{N\text{-}1,d\text{-}1} |\alpha_{n,c}(t)|^2 = 1 \qquad (3\text{-}03)$$

إن عملية اختيار شكل الحالة الابتدائية لها تأثيرها المهم في تحديد السلوك التطوري و النهائي للمشي الكمي (لأن ديناميكا المشي الكمي هو ديناميكا حتمي) عكس المشي الكلاسيكي الذي يكون له دائما نفس السلوك مهما غيرنا اختيارنا للحالة الابتدائية. كما أن الحالة الابتدائية، سواء الكلاسيكية منها أو الكميّة، يمكن أن تكون عبارة عن حالة بسيطة على شكل إحدى حالات أشعة فضاء العقد مع إحدى حالات أشعة فضاء العملة. أي:

$$|\psi(T=0)\rangle = |c,n\rangle = (|c\rangle\otimes|n\rangle) \qquad (3\text{-}04)$$

و يمكن أن تكون عبارة عن حالة غير بسيطة على شكل تركيبة خطية متوازنة أو غير متوازنة، لمجموعة معينة من أشعة الحالات الممكنة، سواء في حالات فضاء العقد أو في حالات فضاء العملة.

$$|\psi(T=0)\rangle = \left(\sum_{c=0}^{\grave{d}\leq d\text{-}1} \alpha_c|c\rangle \otimes \sum_{n=0}^{\grave{N}<N\text{-}1} \alpha_n|n\rangle\right) \quad , \quad \begin{matrix} \sum_{c=0}^{\grave{d}} |\alpha_c|^2 = 1 \\ \sum_{n=0}^{\grave{N}} |\alpha_n|^2 = 1 \end{matrix} \qquad (3\text{-}05)$$

لنختر الحالة البسيطة (لتبسيط فكرة بناء النموذج الكمي). و نقل أن حالة جسيمتنا كانت عند بداية الدراسة في اللحظة $t=0$ (أو بعد الخطوة $T=0$)، في عقدة معينة $|n\rangle$ من فضاء العقد و في (أو ملتفت جهة) أحد أطراف و صلاتها $|c\rangle$ من فضاء العملة [8]:

$$|\psi(T=0)\rangle = |c,n\rangle = (|c\rangle\otimes|n\rangle) \qquad (3\text{-}06)$$

لمحاكاة المشي العشوائي أثناء كل خطوة وفق رياضيات ميكانيك الكم هناك مرحلتين:

مرحلة أولى: نحاكي فيها بقاء الشخص في مكانه و رميه العملة النقدية، لحصوله على كل وُجُوهِهَا الممكنة و باحتمال متساو (أو منحاز جزئيا، المهم ألاّ يكون منحازا كليا لأحد الوجوه الممكنة، للحفاظ

[8] كالإلكترون في موضع معين أو مستوى معين ما، و بسبين علوي أو سفلي مثلا في حالة "$d=2$".

على آلية حدوث العشوائية)، و ذلك عن طريق تطبيق مؤثر **واحدي** جديد ما على شعاع الحالة الابتدائي للجسيمة، حيث يجب أن يعطي شعاع حالة جديد، يبقي الجسيمة في نفس مكانها أو العقدة التي كانت فيها، و لكن مع ظهور جميع حالات فضاء العملة الممكنة و باحتمال متساو (أو شبه متساو). لتصميم هكذا مؤثر واحدي يلبي هذا السلوك يجب أن يستوفي الشروط التالية:

أولا: لكي يبقي الجسيمة في مكانها يكفي ألاّ يؤثر في فضاء العقد و أن يكون مستقلا عنه.

ثانيا: لكي يظهر جميع حالات فضاء العملة الممكنة يكفي أن يتخذ أشعة وحدة هذا الفضاء كأشعة ذاتية له.

ثالثا: لكي تظهر جميع حالات فضاء العملة باحتمال متساوي (أو منحاز جزئيا) يكفي أن تتساوى (أو تختلف قليلا) جميع مربع طويلة عناصر أعمدة مصفوفته فيما بينها مثنى مثنى على الترتيب.

رابعا: لكي لا يكون منحازا بالكلية لأي أحد من أشعته الذاتية (الحفاظ على آلية حدوث العشوائية). يكفي أن يخلو كل عمود من مصفوفته من عنصر واحد، يَسْتَفْرِدُ مربع طويلته بالوحدة "1" (و التالي البقية حتما ستكون معدومة).

كل مؤثر يستوفي هذه الشروط سنسميه بمؤثر العُمْلَة "$\mathbb{C}$" (نسبة لدوره المماثل لدور العملة النقدية في الحالة الكلاسيكية). و سيؤدي تطبيقه على شعاع الحالة الابتدائية إلى الحفاظ على أشعة وحدة فضاء العقد بدون تغير مع ظهور تركيبة خطية لجميع أشعة وحدة فضاء العملة. تتعلق معاملاتها برتبة شعاع وحدة فضاء العملة $\langle c|$ الذي طبق عليه مؤثر العملة $\mathbb{C}$. أي:

$$[\mathbb{C}_d \otimes \mathbb{1}_N]\,(|c\rangle \otimes |n\rangle) = \sum_{i=0}^{d-1} (\beta_i)_c (|i\rangle \otimes |n\rangle) \qquad , \qquad \sum_{i=0}^{d-1} |(\beta_i)_c|^2 = 1 \quad (3\text{-}07)$$

حيث أن المعاملات "$(\beta_i)_c$" هي عناصر العمود ذو الرتبة c من مصفوفة مؤثر العملة $\mathbb{C}$.

مرحلة ثانية: نحاكي فيه تحديد اتجاه خطوات الشخص الممكنة بحسب نتائج عملية رمي العملة النقدية، و ذلك عن طريق تطبيق مؤثر واحدي جديد يسمونه بمؤثر القفزات (shift operator) الواحدي "$\mathbb{S}$"، يعمل على نقل شعاع حالة الجسيمة إلى مختلف العقد المتصلة و وفق أطراف وصلاتها المناسبة، و ذلك بحسب كل حالة من حالات العملة التي ظهرت بعد تطبيق مؤثر العملة. عمليا هذا السلوك هو نفسه سلوك دالة توزع ترقيم العقد و أطرافها e. و منه يمكننا أن نستنتج مباشرة العبارة العامة

لمؤثر القفزات $\mathbb{S}$:

$$\mathbb{S}\cdot(|a\rangle\otimes|n\rangle) = |e(a,n)\rangle = |(b,v)\rangle = |b\rangle\otimes|v\rangle \qquad (3\text{-}08\text{-أ})$$

$$\mathbb{S}\cdot\big(\mathbb{S}\cdot(|a\rangle\otimes|n\rangle)\big) = \big|e\big(e(a,n)\big)\big\rangle = |a\rangle\otimes|n\rangle \qquad (3\text{-}08\text{-ب})$$

المعادلة الأخيرة تبين أن المؤثر $\mathbb{S}$ (الموافق لخطوة الذهاب) يساوي إلى مقلوب نفسه (الموافق إلى خطوة الإياب) و بالتالي فهو واحدي ؟ عادة إذا كانت $a{=}b$ في المعادلة (3-08-أ) يسمى المؤثر $\mathbb{S}$ بمؤثر القفزات، و لكن إذا كانت $a{\neq}b$ فيطلق على مؤثر القفزات بمؤثر الشقلبة (flip-flop operator).

بجمع تأثير هذين المؤثرين مع مراعاة ترتيب مرحلة تأثيرهما، يمكننا استخلاص عبارة مؤثر التطور أو المشي للخطوة الأولى:

$$\big[\mathbb{S}\,.\,[\mathbb{C}\otimes\mathbb{1}_N]\big]\cdot(|c\rangle\otimes|n\rangle) = \mathbb{U}\cdot|\psi(T{=}0)\rangle = |\psi(T{=}1)\rangle \qquad (3\text{-}09)$$

أو لكل خطوة منفردة:

$$|\psi(T{+}1)\rangle = \mathbb{U}\,|\psi(T)\rangle = \big[\mathbb{S}\,.\,[\mathbb{C}\otimes\mathbb{1}_N]\big]\,\cdot\,|\psi(T)\rangle \qquad (3\text{-}10)$$

و بالتالي، إيجاد معادلة المشي من أجل أي عدد معين من الخطى T:

$$|\psi(T)\rangle = \big[\mathbb{S}\,.\,[\mathbb{C}\otimes\mathbb{1}_N]\big]^T\,\cdot\,(|R\rangle\otimes|n) \qquad (3\text{-}11)$$
$$|\psi(T)\rangle = \,[\mathbb{U}]^T\,\cdot\,|\psi(0)\rangle$$

ψ: شعاع الحالة الذي يصف حالة الجسيمة. و يتكون من $(d{\cdot}N)$ مركبة، الموافقة لـ N عقدة من فضاء هيلبرت مُنْحَلَتاً لـ d درجة بِفِعل بُعْد فضاء مؤثر العملة.

$\psi(0)$: هو شعاع الحالة في اللحظة $t=0$ أو بعد الخطوة $T=0$.

$\psi(T)$: هو شعاع الحالة بعد T خطوة معينة أو مختارة.

$\mathbb{S}$: مؤثر القفزات. و هو مصفوفة مربعة واحدية $\big((d{\cdot}N){\times}(d{\cdot}N)\big)$ صيغتها التفصيلية ستكون بحسب اختيارنا لتوزيع ترقيم العقد وأطراف وصلاتها (سواء كان ممنهجا أم لا).

$\mathbb{C}$: مؤثر العُمْلَة، و هو عبارة عن أي مصفوفة واحدية مربعة $(d{\times}d)$ شريطة أن لا تكون نتائج تأثيرها منحازة "بالكلية" لأحد أشعتها الذاتية (لمحاكات فكرة العشوائية).

$\mathbb{U}$: مؤثر التطور، و هو مصفوفة مربعة واحدية $\big((d{\cdot}N){\times}(d{\cdot}N)\big)$.

[9] لكن يبقى الشرط الكافي، لواحدية مؤثر القفزات في الحالة العامة، هو إتباعه لتوزيع الترقيم الذي يتخذه مخططه (ذو البنية المنتظمة).

بعد تكرار المشي لعدة خطوات T سيتغير احتمال تواجد الجسيمة عبر جميع العقد بعدما كان في البداية كله متمركزا في العقدة الابتدائية المختارة فقط، حيث يمكننا أن نجد احتمال تواجد الجسيمة عند كل عقدة. أولاً بتطبيق مؤثر القياس بالنسبة لكل عقدة على حدى $\mathbb{M}_n$ على مركبات شعاع الحالة $\psi(T)$، و ذلك للحصول على شعاع الحالة الجزئي $\psi_n(T)$ (مركباته كلها معدومة باستثناء المركبات المنحلة الموافقة للعقدة n من شعاع الحالة الكلي $\psi(T)$). أي:

$$\begin{aligned}\psi_n(T) &= \mathbb{M}_n \cdot \psi(T)\\ \psi_n(T) &= [\mathbb{1}_d \otimes |n\rangle\langle n|] \cdot \psi(T)\end{aligned} \tag{3-12}$$

و ثم ثانيا، يمكننا أن نحسب الاحتمال الجزئي لتواجد الجسيمة عند كل عقدة معينة n من مجموع العقد، عن طريق الجداء السلمي لشعاع الحالة الجزئي $\psi_n(T)$ مع نفسه.

$$P_n(T) = \langle \psi_n(T) | \psi_n(T) \rangle \tag{3-13}$$

و أخيرا بناء شعاع الاحتمال الكلي عن طريق مركباته (الاحتمالات الجزئية) بعد كل عدد معين من الخطوات T.

$$P(T) = \begin{pmatrix} P_{-E}(T) = \langle \psi(0)^+ \cdot \mathbb{U}^{+^T} \cdot \mathbb{M}_{-E}^+ | \mathbb{M}_{-E} \cdot \mathbb{U}^T \cdot \psi(0) \rangle \\ \vdots \\ P_{-1}(T) = \langle \psi(0)^+ \cdot \mathbb{U}^{+^T} \cdot \mathbb{M}_{-1}^+ | \mathbb{M}_{-1} \cdot \mathbb{U}^T \cdot \psi(0) \rangle \\ P_0(T) = \langle \psi(0)^+ \cdot \mathbb{U}^{+^T} \cdot \mathbb{M}_0^+ | \mathbb{M}_0 \cdot \mathbb{U}^T \cdot \psi(0) \rangle \\ P_1(T) = \langle \psi(0)^+ \cdot \mathbb{U}^{+^T} \cdot \mathbb{M}_1^+ | \mathbb{M}_1 \cdot \mathbb{U}^T \cdot \psi(0) \rangle \\ \vdots \\ P_E(T) = \langle \psi(0)^+ \cdot \mathbb{U}^{+^T} \cdot \mathbb{M}_E^+ | \mathbb{M}_E \cdot \mathbb{U}^T \cdot \psi(0) \rangle \end{pmatrix} \tag{3-14}$$

عمليا يمكننا القول أننا أنتهينا من بناء النموذج الرياضياتي للمشي العشوائي الكمي، و يمكننا الآن الشروع في استعمال معادلات هذا النموذج على بعض الأمثلة الشهيرة و المهمة، لتوضيح فكرة المشي الكمي أكثر و التعرض لأهم الفروقات بين نتائجه و نتائج المشي الكلاسيكي.

2.3 البنى الغير منتهية كالخط المستمر (ذات الاتجاهين):

1.2.3 معادلات المشي العشوائي:

في حالة المخطاط ذو الاتجاهين المبين في الوثيقة 3-01 لدينا N عقدة و كل عقدة لها وصلتين (أي d=2). سنحافظ على التمثيل الشعاعي للعقد الذي استخدمناه في فصل المشي الكلاسيكي لتمثيل

الحالات $|n\rangle$. و بنفس الطريقة أو الفكرة سنعبر عن مختلف أطراف الوصلات بأشعة فضاء العملة ذو البعدين. أي:

$$|0\rangle \equiv |R\rangle = \begin{pmatrix}1\\0\end{pmatrix} \quad , \quad |1\rangle \equiv |L\rangle = \begin{pmatrix}0\\1\end{pmatrix} \tag{3-15}$$

لدراسة المشي الكمي عبر هذا المخطاط سنختار أبسط حالة ابتدائية، حيث أن المتحرك سيكون مثلا قبل الانطلاق في المشي و باحتمال كلي في الحالة:

$$|\psi(T{=}0)\rangle = (|R\rangle \otimes |0\rangle) \tag{3-16}$$

العبارة التفصيلية لمصفوفة مؤثر العملة هناك ثلاث أنواع شهيرة مستعملة (بدلالة بعدها d):

1) هادمارد $\mathbb{H}_d$ (Hadamard)، دوما معتدل:

$$\begin{aligned} \mathbb{H}_1 &= [1] \qquad & \mathbb{H}_2 &= \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & \text{-}1\end{bmatrix} \\ \mathbb{H}_{2^d} &= \mathbb{H}_2 \otimes \mathbb{H}_{2^{d\text{-}1}} = \begin{bmatrix}\mathbb{H}_{2^{d\text{-}1}} & \mathbb{H}_{2^{d\text{-}1}} \\ \mathbb{H}_{2^{d\text{-}1}} & \text{-}\mathbb{H}_{2^{d\text{-}1}}\end{bmatrix} \text{ for } d \geq 2 \end{aligned} \tag{3-17}$$

2) تحويل فوريه المباشر $\mathbb{F}_d$ (DFT)، دوما معتدل:

$$\mathbb{F}_d = \frac{1}{\sqrt{d}}\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{d-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(d-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{d-1} & \omega^{2(d-1)} & \cdots & \omega^{(d-1)^2} \end{pmatrix} \quad \text{and } \omega = e^{\text{-}i2\pi/d} \tag{3-18}$$

3) جروفر $\mathbb{G}_d$ (Grover)، منحاز جزئيا:

$$\mathbb{G}_d = \frac{1}{d}\begin{pmatrix} 2-d & 2 & \cdots & 2 \\ 2 & 2-d & \cdots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ 2 & 2 & 2 & 2-d \end{pmatrix} \tag{3-19}$$

في حالة $d{=}2$. نجد أن مصفوفة مؤثر جروفر $\mathbb{G}_2$ سوف تكون منحازة بالكلية (عنصر واحد مربع طويلته يستحوذ على الوحدة 1 في كل عمود)، أي:

$$\mathbb{G}_2 = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \tag{3-20}$$

و بالتالي لن نستخدمه لأنه لن يخدمنا في إنتاج العشوائية. أما مؤثر فوريه $\mathbb{F}_2$ فمصفوفته ستكون مساوية تماما لمصفوفة هادمارد $\mathbb{H}_2$، حيث ستكون عبارته الشعاعية على النحو التالي:

$$\mathbb{H}_2 = \frac{1}{\sqrt{2}}\left((|R+L\rangle\langle R|\otimes|n\rangle\langle n|) + (|R\text{-}L\rangle\langle L|\otimes|n\rangle\langle n|)\right) \quad (3\text{-}21)$$

و الذي يمتاز بخاصية "الاعتدال التام" حيث كل نتائجه الممكنة لها نفس احتمال الظهور. و بالتالي فهو يحاكي العملة النقدية في الحالة الكلاسيكية من حيث تساوي نتائجها بالضبط.

مؤثر القفزات $\mathbb{S}$ (المعادلة 08-3) ستكون عبارته الشعاعية موافقة تماما لتوزيع الترقيم الذي يتخذه مخططاه في الحالات "أ-أ" ، "ب" و "أ-ب" من الوثيقة 01-03 على الترتيب:

$$\mathbb{S} = \left(|L\rangle\langle R|\otimes\left(\sum_{n=-E}^{E\text{-}1}|n{+}1\rangle\langle n| + |\text{-}E\rangle\langle E|\right)\right) + \left(|R\rangle\langle L|\otimes\left(\sum_{n=-E+1}^{E}|n\text{-}1\rangle\langle n| + |E\rangle\langle\text{-}E|\right)\right) \quad (3\text{-}22\text{-أ})$$

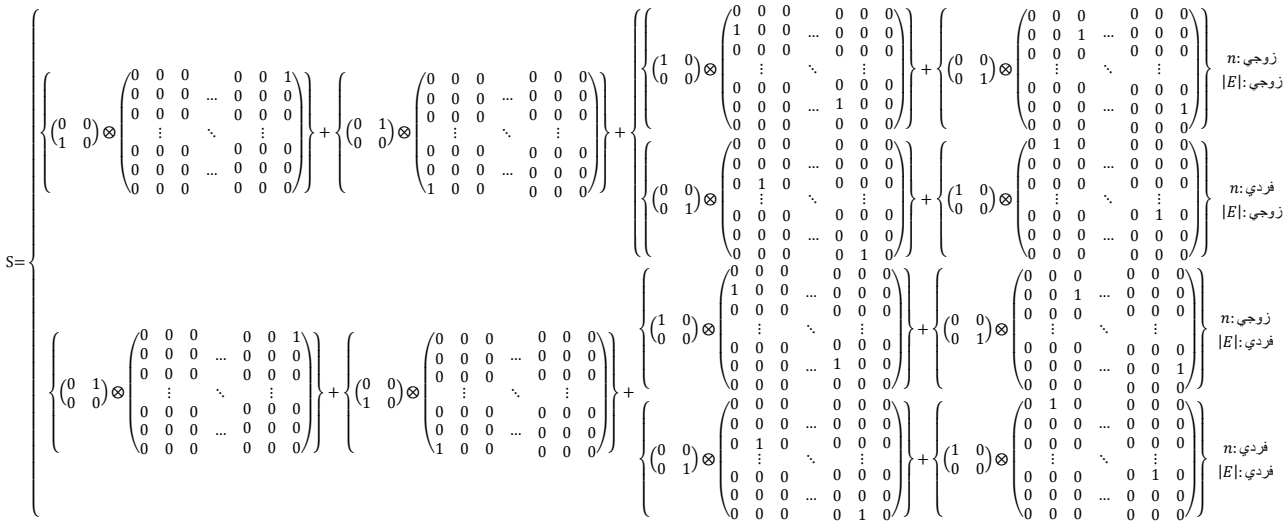
$$\mathbb{S} = \left(|R\rangle\langle R|\otimes\left(\sum_{n=-E}^{E\text{-}1}|n{+}1\rangle\langle n| + |\text{-}E\rangle\langle E|\right)\right) + \left(|L\rangle\langle L|\otimes\left(\sum_{n=(-E+1)}^{E}|n\text{-}1\rangle\langle n| + |E\rangle\langle\text{-}E|\right)\right) \quad (3\text{-}22\text{-ب})$$

$$\mathbb{S} = \begin{cases} (|L\rangle\langle R|\otimes|\text{-}E\rangle\langle E|) + (|R\rangle\langle L|\otimes|E\rangle\langle\text{-}E|) + \begin{cases} \left(|R\rangle\langle R|\otimes\sum_{n=-E}^{E\text{-}1}|n{+}1\rangle\langle n|\right) + \left(|L\rangle\langle L|\otimes\sum_{n=(-E+1)}^{E}|n\text{-}1\rangle\langle n|\right) & n: \text{زوجي} \\ & |E|: \text{زوجي} \\ \left(|L\rangle\langle L|\otimes\sum_{n=-E}^{E-1}|n{+}1\rangle\langle n|\right) + \left(|R\rangle\langle R|\otimes\sum_{n=(-E+1)}^{E}|n\text{-}1\rangle\langle n|\right) & n: \text{فردي} \\ & |E|: \text{زوجي} \end{cases} \\ (|R\rangle\langle L|\otimes|\text{-}E\rangle\langle E|) + (|L\rangle\langle R|\otimes|E\rangle\langle\text{-}E|) + \begin{cases} \left(|R\rangle\langle R|\otimes\sum_{n=-E}^{E\text{-}1}|n{+}1\rangle\langle n|\right) + \left(|L\rangle\langle L|\otimes\sum_{n=(-E+1)}^{E}|n\text{-}1\rangle\langle n|\right) & n: \text{زوجي} \\ & |E|: \text{فردي} \\ \left(|L\rangle\langle L|\otimes\sum_{n=-E}^{E\text{-}1}|n{+}1\rangle\langle n|\right) + \left(|R\rangle\langle R|\otimes\sum_{n=(-E+1)}^{E}|n\text{-}1\rangle\langle n|\right) & n: \text{فردي} \\ & |E|: \text{فردي} \end{cases} \end{cases} \quad (3\text{-}22\text{-ج})$$

و عباراتهم المصفوفية هي أيضا على الترتيب:

$$\mathbb{S} = \left\{\begin{pmatrix}0&0\\1&0\end{pmatrix}\otimes\begin{pmatrix}0&0&0&&0&0&1\\1&0&0&\cdots&0&0&0\\0&1&0&&0&0&0\\&\vdots&&\ddots&&\vdots&\\0&0&0&&0&0&0\\0&0&0&\cdots&1&0&0\\0&0&0&&0&1&0\end{pmatrix}\right\} + \left\{\begin{pmatrix}0&1\\0&0\end{pmatrix}\otimes\begin{pmatrix}0&1&0&&0&0&0\\0&0&1&\cdots&0&0&0\\0&0&0&&0&0&0\\&\vdots&&\ddots&&\vdots&\\0&0&0&&0&1&0\\0&0&0&\cdots&0&0&1\\1&0&0&&0&0&0\end{pmatrix}\right\} \quad (3\text{-}23\text{-أ})$$

$$\mathbb{S} = \left\{\begin{pmatrix}1&0\\0&0\end{pmatrix}\otimes\begin{pmatrix}0&0&0&&0&0&1\\1&0&0&\cdots&0&0&0\\0&1&0&&0&0&0\\&\vdots&&\ddots&&\vdots&\\0&0&0&&0&0&0\\0&0&0&\cdots&1&0&0\\0&0&0&&0&1&0\end{pmatrix}\right\} + \left\{\begin{pmatrix}0&0\\0&1\end{pmatrix}\otimes\begin{pmatrix}0&1&0&&0&0&0\\0&0&1&\cdots&0&0&0\\0&0&0&&0&0&0\\&\vdots&&\ddots&&\vdots&\\0&0&0&&0&1&0\\0&0&0&\cdots&0&0&1\\1&0&0&&0&0&0\end{pmatrix}\right\} \quad (3\text{-}23\text{-ب})$$



(ج-3-23)

ومنه بتطبيق عبارة مؤثر التطور $\mathbb{U}_{\mathbb{H}_2}$ (المعادلة 09-3) من أجل الحصول على دالة الحالة الموافقة للخطوة الأولى نجد:

$$\begin{aligned}\mathbb{U}_{\mathbb{H}_2}\,|\psi(T{=}0)\rangle &= \left[\mathbb{S}\cdot\ [\mathbb{H}_2\otimes\mathbb{1}_N]\right](|R\rangle\otimes|0\rangle)\\ &= \frac{1}{\sqrt{2}}[|L\rangle\otimes|1\rangle + |R\rangle\otimes|\text{-}1\rangle]\\ &= |\psi(T{=}1)\rangle\end{aligned} \tag{3-24}$$

أين أثرنا على الحالة الابتدائية بمؤثر العملة $\mathbb{H}$ ثم بمؤثر القفزات $\mathbb{S}$ على الترتيب. أي:

$$\begin{aligned}(|R\rangle\otimes|0\rangle) &\xrightarrow{\mathbb{H}_2\otimes\mathbb{1}_N} \frac{1}{\sqrt{2}}[(|R\rangle{+}|L\rangle)\otimes|0\rangle]\\ &\xrightarrow{\mathbb{S}} \frac{1}{\sqrt{2}}[(|L\rangle\otimes|1\rangle) + (|R\rangle\otimes|\text{-}1\rangle)]\end{aligned} \tag{3-25}$$

هنا يمكن أن نميز سلوكين حاسمين و هامين هما:

أولاً: إذا قمنا **بعملية القياس** بعد هذه الخطوة الأولى سنجد أن الجسيمة تتواجد في العقدتين 1 و 1- بنفس الاحتمال و هو $\left|\frac{1}{\sqrt{2}}\right|^2$ (أي كل حالة ستأخذ نصف قيمة احتمال الحالة التي تطورت منها قبل تنفيذ الخطوة). و لكن الشيء المهم هنا أنّ بعد عملية القياس ستصبح الحالتان الناتجتان بعد الخطوة الأولى $\{(|L\rangle\otimes|1\rangle)$، $(|R\rangle\otimes|\text{-}1\rangle)\}$ منفصلتين تماما عن بعضهما البعض. حيث كل حالة ستشكل نظام جزئي موضعي منفصل عن الأخرى ضمن النظام الكلي (و هو تطور الحالة الابتدائية التي انطلق منها المشي العشوائي). مما يعني أن في الخطوة التي ستليها (أي الخطوة الثانية) سنطبق مؤثر التطور $\mathbb{U}_{\mathbb{H}_2}$ على كل حالة على "**حدى**" (أي: نطبق $\mathbb{U}_{\mathbb{H}_2}$ على $(|L\rangle\otimes|1\rangle)$، و نطبق $\mathbb{U}_{\mathbb{H}_2}$ على

($|R\rangle\otimes|\text{-}1\rangle$) و بالتالي إذا أستمررنا في إجراء عملية القياس بعد كل خطوة سنحصل دائمًا على حالتين **"منفصلتين"** لهما **"نصف"** احتمال الحالة التي **"تطورَتا منها"**، و هذا سيعطي في النهاية بالضبط التوزيع الكلاسيكي للمشي العشوائي الذي تعرضنا له في الفصل السابق.

ثانياً: إذا لم نقم بعملية القياس بعد كل خطوة سنحافظ بالتالي على الترابط الكمي (quantum correlation) بين الحالات الناتجة لمختلف العقد بعد كل خطوة. مما سيسمح دائمًا؛ بتداخل حالاتها (المعبرة عن مختلف مواضع العقد) و الناتجة خلال كل الخطوات أثناء المشي العشوائي. هذا التداخل هو بالضبط ما سيؤدي إلى تَمَايُزِ المشي الكمي و بشكل جذري عن المشي الكلاسيكي.

2.2.3 الحساب اليدوي و كيفية تحرير البرنامج الحاسوبي:

لتوضيح الفكرة أكثر سنستعرض كيفية تطور شعاع الحالة (حساب يدوي) لبعض من الخطوات الأولى إلى غاية أول ظهور لفرق بين المشي الكمي و الكلاسيكي (سنجده بعد الخطوة الثالثة). أي:

$$
\begin{aligned}
&|\psi(T=0)\rangle \xrightarrow{\mathbb{U}_{\mathbb{H}_2}} |\psi(T=1)\rangle = \frac{1}{\sqrt{2}}\left[(|L\rangle\otimes|1\rangle) + (|R\rangle\otimes|\text{-}1\rangle)\right] \\
&|\psi(T=1)\rangle \xrightarrow{\mathbb{U}_{\mathbb{H}_2}} |\psi(T=2)\rangle = \frac{1}{2}\left[(|L\rangle\otimes|2\rangle) + (|(L\text{-}R)\rangle\otimes|0\rangle) + (|R\rangle\otimes|\text{-}2\rangle)\right] \\
&|\psi(T=2)\rangle \xrightarrow{\mathbb{U}_{\mathbb{H}_2}} |\psi(T=3)\rangle = \frac{1}{2\sqrt{2}}\left[(|L\rangle\otimes|3\rangle) - (|R\rangle\otimes|1\rangle) + (|(L\text{-}2R)\rangle\otimes|\text{-}1\rangle) + (|R\rangle\otimes|\text{-}3\rangle)\right]
\end{aligned}
\qquad (3\text{-}26)
$$

ثم بلغة الاحتمالات (أي مربع طويلة معامل كل حالة) سنحسب و نلخص نتائج احتمالات تواجد الجسيم عبر مختلف العقد من أجل الأربع خطوات الأولى كالتالي:

$-E$	⋯	-4	-3	-2	-1	0	1	2	3	4	⋯	E	n / $P(\mathrm{T})$
0	⋯	0	0	0	0	**1**	0	0	0	0	⋯	0	$P(0)$
0	⋯	0	0	0	**1/2**	**0**	**1/2**	0	0	0	⋯	0	$P(1)$
0	⋯	0	0	**1/4**	**0**	**2/4**	**0**	**1/4**	0	0	⋯	0	$P(2)$
0	⋯	0	**1/8**	**0**	**5/8**	**0**	**1/8**	**0**	**1/8**	0	⋯	0	$P(3)$
0	⋯	**1/16**	**0**	**10/16**	**0**	**2/16**	**0**	**2/16**	**0**	**1/16**	⋯	0	$P(4)$

من أجل عدد كبير من الخطوات سنستعين بمعادلات المشي الكمي التي بنينا بها نموذج المشي الكمي. و تحريرها في شكل برنامج حاسوبي (الملحق: البرنامج 1).

3.2.3 تحليل المنحنيات و استخلاص النتائج:

سنجد أن المشي الكمي سيعطي نتائج مغايرة تماما للمشي الكلاسيكي (الوثيقة02-3) حيث أن:

1) قِمَّةُ احتماله (ترافقها قمم أخرى أقل منها و تليها في الحجم) نجدها تبتعد عن موضع الانطلاق كلما زدنا عدد الخطوات و بمسافة تتناسب طرديا و عدد الخطوات قدرها $\approx \frac{T}{\sqrt{2}}$، عكس المشي الكلاسيكي الذي تبقى قِمَّتُهُ و القمم التي تليها في الحجم دائما متمركزة في موضع انطلاقه و الجوار الذي يليها.

2) تتشتت مركبات الاحتمال الكلي (أي الاحتمالات الجزئية) عن طريق التباين و الانحراف المعياري. لتتناسب بالتقريب مع مربع عدد الخطوات و عدد الخطوات على الترتيب [8]. أي:

$$\sigma^2(T) \propto T^2 \Rightarrow \sigma(T) \propto T \qquad (3\text{-}27)$$

عكس المشي الكلاسيكي التي وجدناها تتناسب بالتقريب مع عدد الخطوات و الجذر التربيعي لعدد الخطوات على الترتيب (المعادلة (08-2)).

مما يعني في المحصلة، و كنتيجة مباشرة و مهمة: أن المشي الكمي **أسرع تربيعيا** في الابتعاد عن موضع انطلاقه مقارنة بالمشي الكلاسيكي.

3.3 البنى الغير منتهية كالخط المستمر (ذو الاتجاهات الثلاثة):

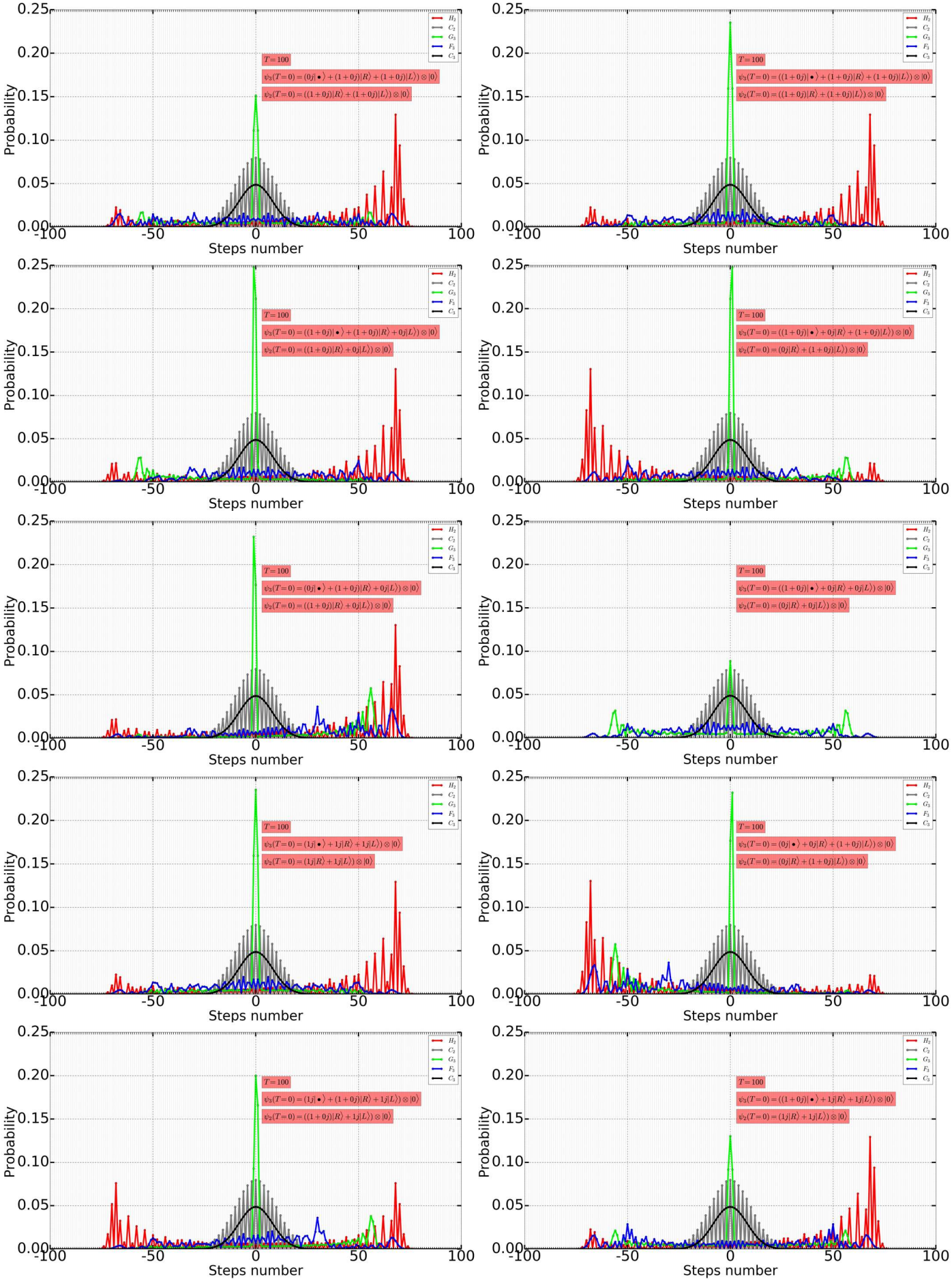
1.3.3 معادلات المشي العشوائي:

في حالة المخطاط ذو الثلاثة اتجاهات سنضيف فقط خطوة الاستراحة. حيث سنحافظ على مجمل معادلات المخطاط ذو الاتجاهين مع تغيير بُعد فضاء العملة ليصبح "$d=3$"، حيث سيكون لدينا ثلاث وصلات ومنه ثلاث أشعة حالة من فضاء العملة، هي:

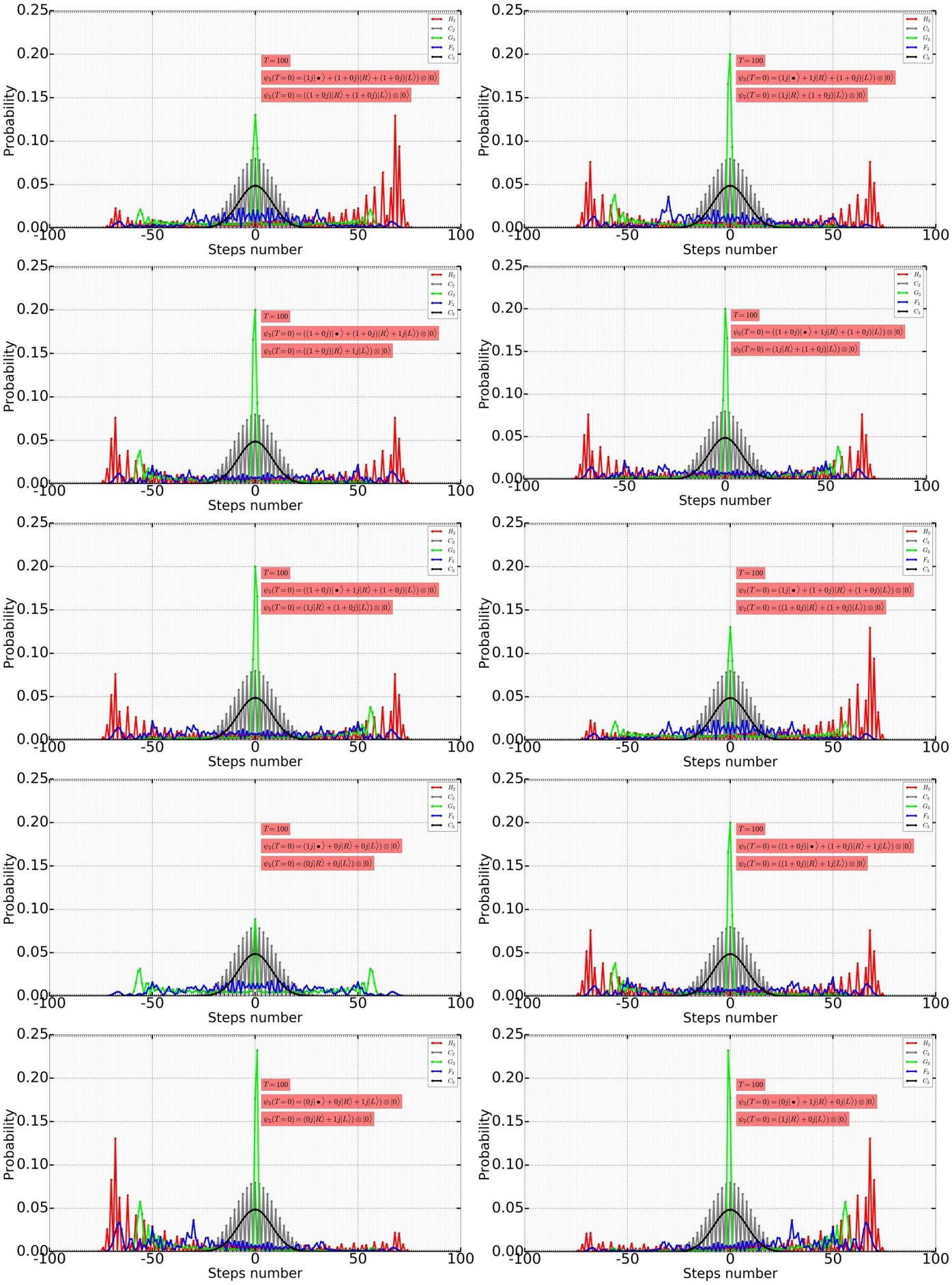
$$|0\rangle \equiv |R\rangle = \begin{pmatrix}1\\0\\0\end{pmatrix}, \qquad |1\rangle \equiv |L\rangle = \begin{pmatrix}0\\1\\0\end{pmatrix}, \qquad |2\rangle \equiv |\bullet\rangle = \begin{pmatrix}0\\0\\1\end{pmatrix} \qquad (3\text{-}28)$$

الحالة الابتدائية، سنختار أبسطها مثل:" $|\psi(T=0)\rangle = (|R\rangle \otimes |0\rangle)$ "

مصفوفة مؤثر العملة ستصبح هي الأخرى ذات ثلاث أبعاد. مصفوفة هادمارد ذات الثلاث أبعاد غير مُعَرَّفَةٍ. و بالتالي سنستعمل فقط مصفوفة مؤثر جروفر $\mathbb{G}_3$:



الوثيقة 02-أ-3: مجموعة من نتائج توزع الاحتمال الكلي $P(T)$ من خلال المشي الكمي (H_2, G_3, F_3) و الكلاسيكي (C_2, C_3) بعد 100 خطوة. و ذلك من أجل مختلف الحالات الابتدائية الممكنة سواء في حالة الاتجاهين (ψ_2) أو الثلاث اتجاهات (ψ_3).



الوثيقة 02-3-ب: تابع للوثيقة 02-2-أ.

$$\mathbb{G}_3 = \frac{1}{3}\begin{pmatrix} -1 & 2 & 2 \\ 2 & -1 & 2 \\ 2 & 2 & -1 \end{pmatrix} \tag{3-29}$$

و مصفوفة فوريه $\mathbb{F}_3$:

$$\mathbb{F}_3 = \frac{1}{\sqrt{3}}\begin{pmatrix} 1 & 1 & 1 \\ 1 & e^{\frac{-i2\pi}{3}} & e^{\frac{-i2\pi}{3}\times 2} \\ 1 & e^{\frac{-i2\pi}{3}\times 2} & e^{\frac{-i2\pi}{3}\times 4} \end{pmatrix} = \frac{1}{\sqrt{3}}\begin{pmatrix} 1 & 1 & 1 \\ 1 & -\frac{1}{2}-\frac{\sqrt{3}}{2}i & -\frac{1}{2}+\frac{\sqrt{3}}{2}i \\ 1 & -\frac{1}{2}+\frac{\sqrt{3}}{2}i & -\frac{1}{2}-\frac{\sqrt{3}}{2}i \end{pmatrix} \tag{3-30}$$

مؤثر القفزات $\mathbb{S}$ عموما سيبقى هو نفسه الذي أستعمل في حالة الخط المفتوح ذو الاتجاهين مع إضافة حد جديد فقط يعبر عن خطوات الاستراحة. و التي ستكون عمليا سهلة في ترميزها، حيث كل خطوة ستحافظ على نفس العقدة و طرف الوصلة التي كانت فيها، أي:

$$|\bullet\rangle\langle\bullet| \otimes \sum_n |n\rangle\langle n| \tag{3-31}$$

و منه تصبح العبارة الشعاعية لمؤثر القفزات $\mathbb{S}$ الذي يصف المخطاط "أ-أ" مثلا هي[10]:

$$\mathbb{S} = \left(|L\rangle\langle R| \otimes \left(\sum_{n=-E}^{E-1} |n+1\rangle\langle n| + |-E\rangle\langle E|\right)\right) + \left(|L\rangle\langle R| \otimes \left(\sum_{n=-E+1}^{E} |n\text{-}1\rangle\langle n| + |E\rangle\langle -E|\right)\right) + \left(|\bullet\rangle\langle\bullet| \otimes \sum_{n=-E}^{E} |n\rangle\langle n|\right) \tag{3-32}$$

و عبارته المصفوفية هي:

$$\mathbb{S} = \left\{\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & & 0 & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & & 0 & 1 & 0 \end{pmatrix}\right\} + \left\{\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & 0 & & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 0 & 0 & & 0 & 0 & 0 \end{pmatrix}\right\}$$
$$+ \left\{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 & & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & & 0 & 0 & 1 \end{pmatrix}\right\} \tag{3-33}$$

و أخيرا، بتطبيق عبارة مؤثر التطور $\mathbb{U}$ (المعادلة 09-3) سنحصل على شعاع الحالة الموافقة لخطوة الأولى.

[10] و بنفس الفكرة سنتعامل مع المخطاط "أ-ب" و "ب" أو أي مخطاط أخر بالنسبة لخطوات الاستراحة.

2.3.3 الحساب اليدوي و كيفية تحرير البرنامج الحاسوبي:

أولا: في حالة استعمال مؤثر جروفر $\mathbb{G}_3$

$$\begin{aligned}\mathbb{U}_{\mathbb{G}_3}|\psi(T{=}0)\rangle &= \left[\mathbb{S}\cdot\ [\mathbb{G}_3\otimes\mathbb{1}_N]\right](|R\rangle\otimes|0\rangle)\\ &= \frac{1}{3}[(\text{-}|L\rangle\otimes|1\rangle)+(2|R\rangle\otimes|\text{-}1\rangle)+(2|\bullet\rangle\otimes|0\rangle)]\\ &= |\psi(T{=}1)\rangle\end{aligned} \tag{3-34}$$

أين أثرنا على الحالة الابتدائية بمؤثر العملة $\mathbb{G}_3$ ثم بمؤثر القفزات $\mathbb{S}$ على الترتيب. أي:

$$\begin{aligned}(|R\rangle\otimes|0\rangle) &\xrightarrow{\mathbb{G}_3\otimes\mathbb{1}_N} \frac{1}{3}[(\text{-}|R\rangle{+}2|L\rangle{+}2|\bullet\rangle)\otimes|0\rangle]\\ &\xrightarrow{\mathbb{S}} \frac{1}{3}[(\text{-}|L\rangle\otimes|1\rangle)+(2|R\rangle\otimes|{-}1\rangle)+(2|\bullet\rangle\otimes|0\rangle)]\end{aligned} \tag{3-35}$$

بلغة الاحتمالات، نجد أن مؤثر جروفر $\mathbb{G}_3$ منحاز جزئيا، حيث سيعطي الحالة التي طبق عليها باحتمال "$\frac{1}{9}$" (أعني أحد حالات العملة التي سيطبق عليها المؤثر. و في المثال الذي استعرضناه أعلاه كانت هي الحالة "$|R\rangle$" من شعاع الحالة الابتدائية التي اخترناها "$(|R\rangle\otimes|0\rangle)$"). و يعطي الحالتين الأخرتين المتبقيتين باحتمال "$\frac{4}{9}$" (في المثال الذي استعرضناه أعلاه كانتا هما الحالتين "$|L\rangle$" و "$|\bullet\rangle$" من فضاء العملة). عمليا يمكن التغلب على هذه الخاصية بتحضير مزيج متكافئ من حالات العملة في دالة الحالة الابتدائية. أي:

$$|\psi(T{=}0)\rangle = \left(\frac{1}{\sqrt{3}}(|R\rangle + |L\rangle + |\bullet\rangle)\right)\otimes|0\rangle \tag{3-36}$$

و بالتالي بعد حساب احتمالات نتائج الخطوة الأولى. نجد أن كل حالة ناتجة ستُعطى باحتمال "$\frac{1}{3}$". و هذه الميزة لمصفوفات جروفر يمكن اعتبارها من العيوب في حالات و من المحاسن في حالات أخرى.

ثانيا: في حالة استعمال فوريه $\mathbb{F}_3$

$$\begin{aligned}\mathbb{U}_{\mathbb{F}_3}|\psi(T{=}0)\rangle &= \left[\mathbb{S}\cdot\ [\mathbb{F}_3\otimes\mathbb{1}_N]\right](|R\rangle\otimes|0\rangle)\\ &= \frac{1}{\sqrt{3}}[(|L\rangle\otimes|1\rangle)+(|R\rangle\otimes|{-}1\rangle)+(|\bullet\rangle\otimes|0\rangle)]\\ &= |\psi(T{=}1)\rangle\end{aligned} \tag{3-37}$$

أين أثرنا على الحالة الابتدائية بمؤثر العملة $\mathbb{F}_3$ ثم بمؤثر القفزات $\mathbb{S}$ على الترتيب. أي:

$$\begin{aligned}(|R\rangle\otimes|0\rangle) &\xrightarrow{\mathbb{F}_3\otimes\mathbb{1}_N} \frac{1}{\sqrt{3}}[(|R\rangle{+}|L\rangle{+}|\bullet\rangle)\otimes|0\rangle]\\ &\xrightarrow{\mathbb{S}} \frac{1}{\sqrt{3}}[(|L\rangle\otimes|1\rangle)+(|R\rangle\otimes|{-}1\rangle)+(|\bullet\rangle\otimes|0\rangle)]\end{aligned} \tag{3-38}$$

عموما فمصفوفات فوريه $\mathbb{F}_d$ تعتبر دوما معتدلة (نفس الاحتمال لجميع نتائجها). مثلها مثل مصفوفات هادمارد $\mathbb{H}_d$. و الفرق الوحيد الذي يميزها هو ظهور الطور (أي $e^{-i\alpha}$) الذي يمكن أن يحدث تغيرات مختلفة بسبب تداخل دوال الحالة أثناء التطور الزمني للمشي الكمي، مقارنة مع التطور الذي يوافق مصفوفات هادمارد.

من أجل عدد كبير من الخطوات سنستعين بمعادلات المشي الكمي، و تحريرها في شكل برنامج حاسوبي (الملحق: البرنامج 1).

3.3.3 تحليل المنحنيات و استخلاص النتائج:

بعد تحليل البيانات (الوثيقة 02-3) يمكن أن نلخص النتائج في ما يلي:

1) كالعادة سنجد أن المشي الكمي سيعطي نتائجا مغايرة تماما للمشي الكلاسيكي حيث أن الاحتمال سينتشر و يمتد مبتعدا عن نقطة انطلاقه عكس الكلاسيكي الذي يبقى متمركزا و محاذيا لنقطة انطلاقه.

2) في حالة جروفر $\mathbb{G}_3$ سيتميز بقمته البارزة و المتمركزة دوما عند نقطة انطلاقه و قمم صغيرة أخرى متناثرة يمينا أحيانا و يسارا أحيانا و متناظرة يمينا و يسارا أحيانا أخرى (و أحيانا قمة واحدة بارزة أو اثنتان) و هذا كله بحسب الحالة الابتدائية المختارة.

3) في حالة فوريه $\mathbb{F}_3$ عموما ليس له قمة بارزة بل قمم متقاربة في الطول تقريبا تمتد مبتعدة عن نقطة انطلاقها (تشبه إلى حد بعيد إشارات تشويش الفضاء). أحيانا منحازة لليمين و أحيانا نحو اليسار و أحيانا أخرى متناظرة بين اليمين و اليسار و ذلك طبعا بحسب الحالة الابتدائية المختارة.

4) إذا قارنا نتائج الخط المستمر ذو الاتجاهين و الثلاثة اتجاهات يمكننا القول أن ذو الاتجاهين يعطي نتائج أفضل في سرعة ابتعاده و كمية الاحتمال التي يحملها معه مقارنة مع ذو الثلاث اتجاهات. و لكن عموما يبقى السلوك الكمي هو الأفضل بكثير في الابتعاد عن نقطة انطلاقه مقارنة بالكلاسيكي.

4.3 البنى المنتهية كالحلقة (ذات الاتجاهين و الاتجاهات الثلاثة):

1.4.3 معادلات المشي العشوائي:

في ما يخص البنى المغلقة سنستخدم نفس المعادلات و التفاصيل التي استعرضناها في حالة البنى

المفتوحة، سواء في حالة الخط ذو الاتجاهين أو ذو الثلاث اتجاهات، مع فرق بسيط و جوهري و هو عدد الخطى الذي سنختاره يجب أن يكون أكبر تماما من نصف عدد عقد المخطاط ناقص واحد (أي: $T > \frac{N-1}{2}$)، و ذلك للحصول على ظاهرة التداخل بين الاحتمال الذي ينتشر من جهة اليمين مع الاحتمال الذي ينتشر من جهة اليسار.

2.4.3 الحساب اليدوي و كيفية تحرير البرنامج الحاسوبي:

الحساب اليدوي و البرنامج (الملحق: البرنامج 1) هو أيضا نفسه الذي استعرضناه في حالة البنى المفتوحة مع مراعاة الشرط $T > \frac{N-1}{2}$ أثناء تنفيذ البرنامج.

3.4.3 تحليل المنحنيات و استخلاص النتائج:

بعد الحصول على البيانات (الوثيقة 03-3) و تحليلها تمكنا من أن نلخص أهم النتائج في ما يلي:

1. السلوك الكمي في البنى المغلقة كالحلقة لا يجنح إلى حالة الاستقرار التي وجدناها في الحالة الكلاسيكية، و هذا حتى من أجل عدد كبير جدا من الخطى T.
2. عموما، السلوك الكمي سواء في حالة الحلقة ذات الاتجاهين أو الثلاثة اتجاهات و لمختلف مؤثرات العملة، يختلف بعضه عن بعض.
3. في حالة مؤثر جروفر $\mathbb{G}_3$ نجد أن قِمَّتُهُ تكون دوما متمركزة في موضع الانطلاق، عكس بقية مؤثرات العملة.

4.3 خاصية الانتقال (الحلقة):

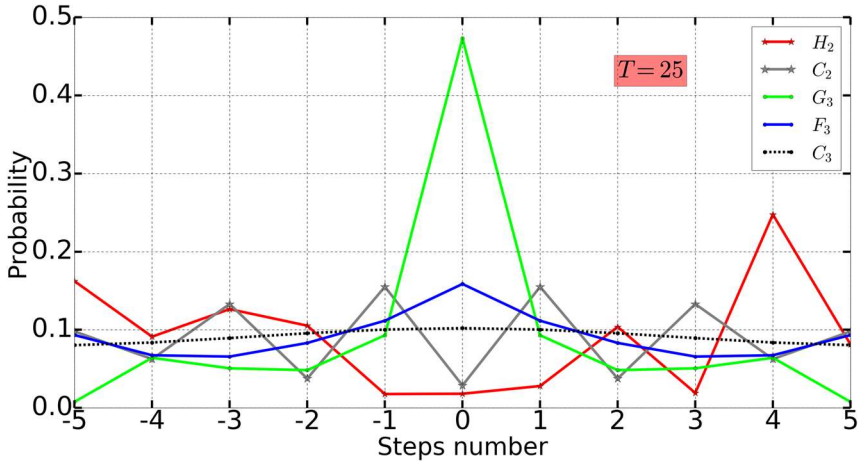
1.4.3 تعريف خاصية الانتقال و شرح كيفية حسابها:

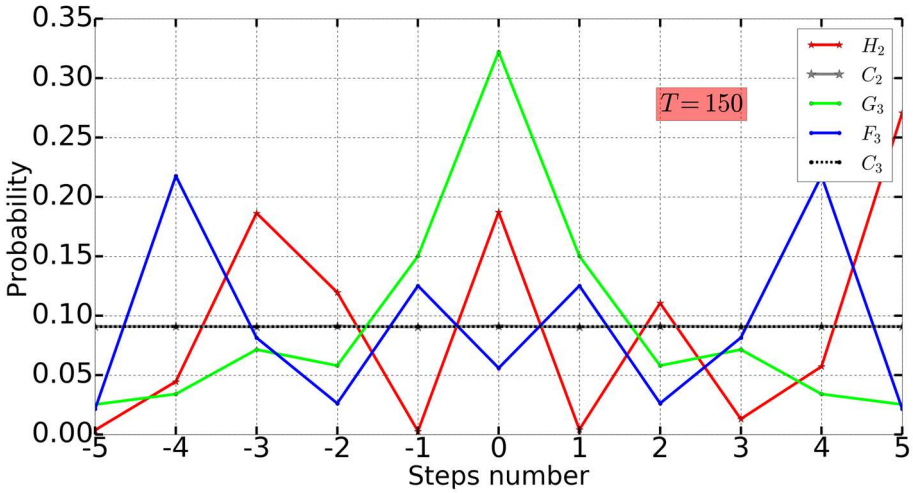
كما فعلنا في المشي الكلاسيكي سنقوم الآن باختبار خاصية التنقل في بنية مغلقة كالحلقة بالنسبة للمشي الكمي، حيث سنحافظ على جميع التعريفات و الأفكار التي أوردناه في حالة المشي الكلاسيكي و لكن مع تغير معادلات الحركة إلى معادلات الحركة الخاصة بالمشي الكمي. حيث ستصبح عبارة الدالة $A(T)$، المسؤولة على انتزاع قيمة الاحتمال عند العقدة الهدف a كالتالي:

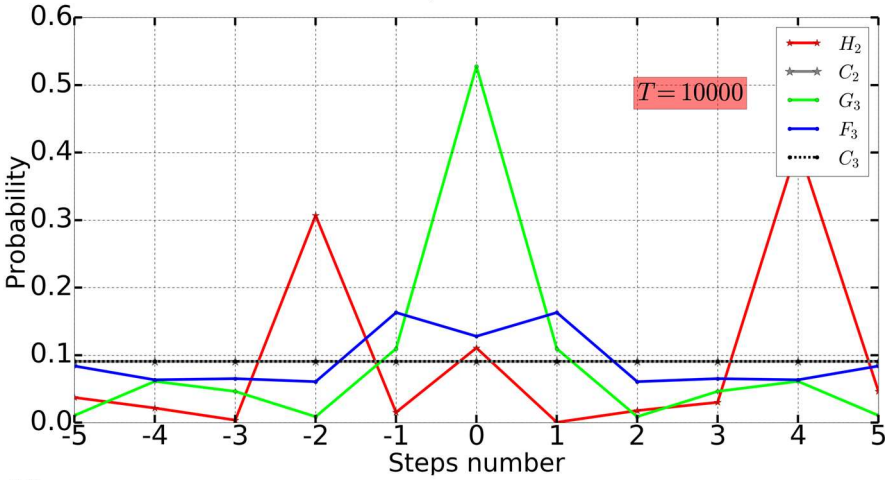
$$A(T) = \sum_{t=0}^{T} \langle \psi_a(t) | \psi_a(t) \rangle = \sum_{t=0}^{T} P_a(t) \qquad (3\text{-}39)$$

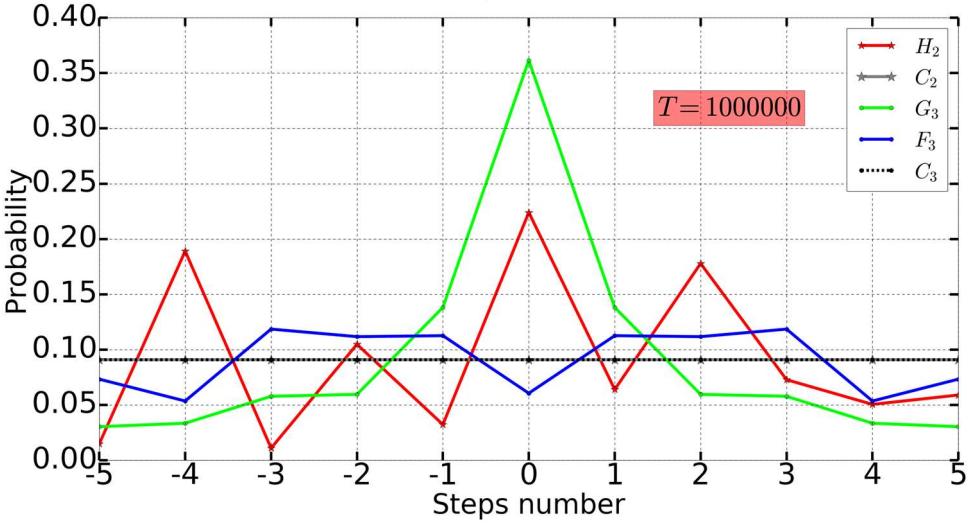
الوثيقة 03-3:

التمثيل البياني لكيفية تطور احتمال تواجد المتحرك. بعد انطلاقه من العقدة 0. عبر مختلف العقد (11 عقدة) لمخطاط على شكل حلقة. في حالة المشي الكلاسيكي: ذو الاتجاهين (C_2)، ذو الثلاثة اتجاهات (C_3). أما في حالة المشي الكمي: هادمارد ذو الاتجاهين (H_2). جروفر في حالة ثلاثة اتجاهات $(\mathbb{G}_3)$. فوريه في حالة ثلاثة اتجاهات $(\mathbb{F}_3)$.

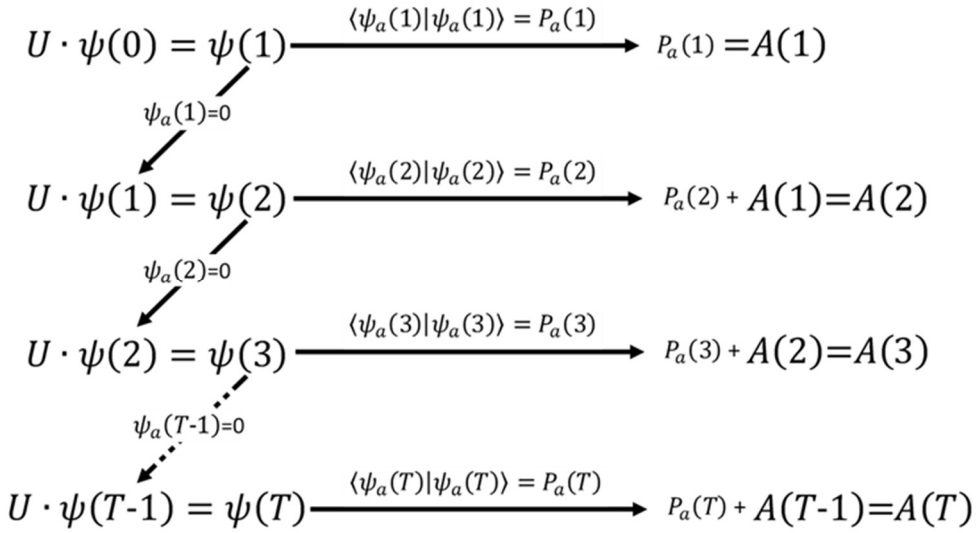








الوثيقة 3-04 توضح فكرة حساب خاصية الانتقال في شكل مخطط توضيحي عن طريق المعادلات و خطوة بعد خطوة. و بالتالي يمكننا في النهاية من حساب تطور الدالة $A(T)$ في حالة المشي الكمي و مقارنتها مع المشي الكلاسيكي.



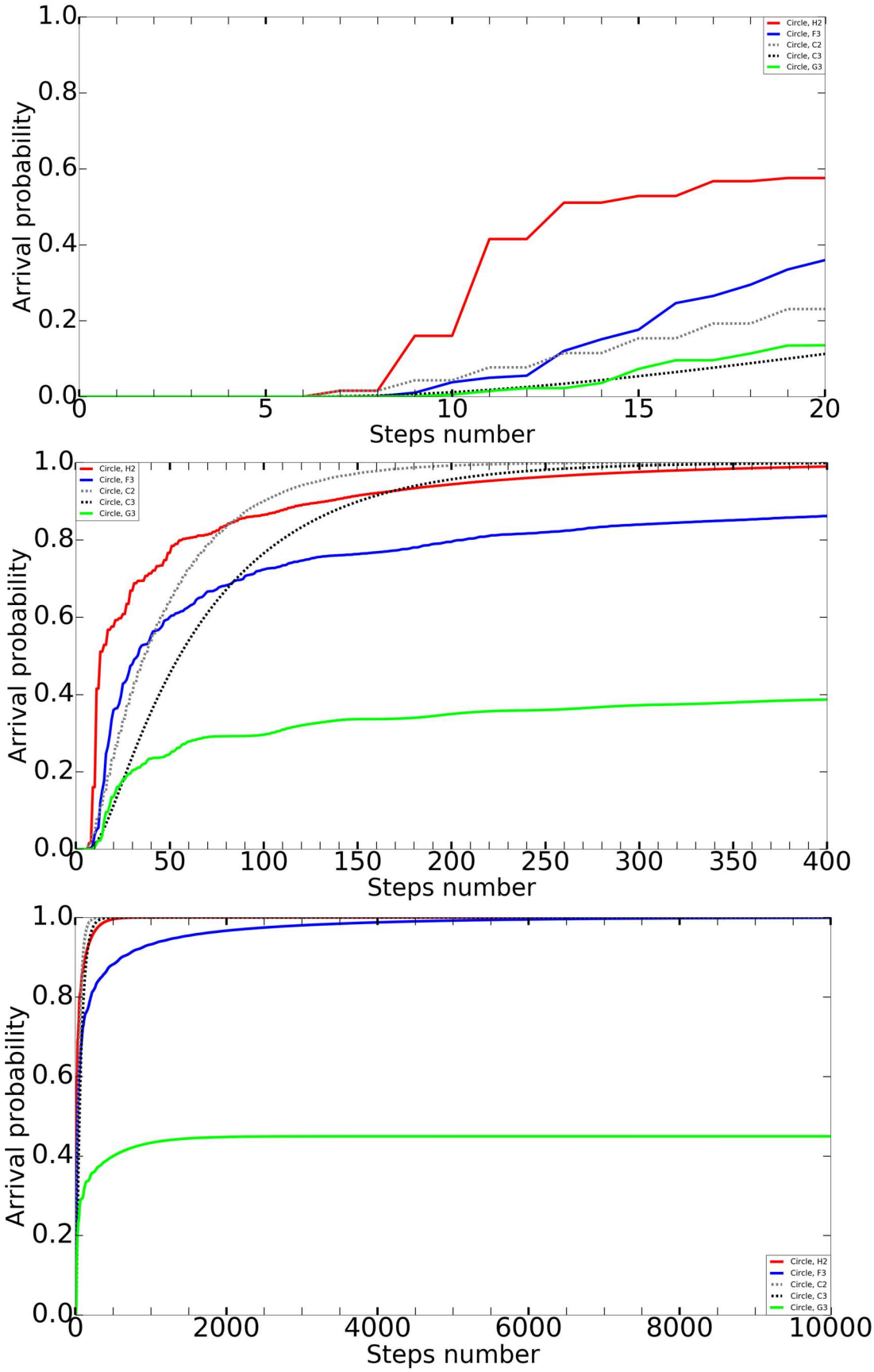
الوثيقة 3-04: مخطط توضيحي، يظهر عن طريق المعادلات كيفية حساب و تطور الاحتمال الممتص $A(T)$ عند موضع الهدف (العقدة a)، إنطلاقا من الخطوة 0 خطوة بخطوة إلى غاية الخطوة T.

2.4.3 البرنامج الحاسوبي، تحليل المنحنيات و استخلاص النتائج:

بعد تنفيذ البرنامج (الملحق: البرنامج 2) حسب فكرة المخطط الموضح في الوثيقة 3-04، نجد أن النتائج (الوثيقة 3-05) توضح جليا أن:

1) المشي الكمي في حالة H_2 يكون في البداية هو الأسرع في نقل الاحتمال (يصل الى أكثر من 80% خلال الـ 85 خطوة الأولى). ثم بعد ذلك يسبقه المشي الكلاسيكي C_2 و C_3 بعد الخطوة 90 و 180 على الترتيب. و يصلان إلى نقل كل الاحتمال الكلي بعد 220 و 320 على الترتيب. بينما المشي الكمي H_2 لا يصل الى نقل كل الاحتمال الكلي الا بعد الخطوة 500.

2) المشي الكمي في حالة F_3 هو الآخر في البداية يكون أسرع من المشي الكلاسيكي (ينقل أكثر من 50% خلال 40 خطوة الأولى) و لكن أقل دائما من المشي الكمي H_2. ثم بعد ذلك يسبقه المشي الكلاسيكي C_2 و C_3 بعد الخطوة 40 و 85 على الترتيب. و لا يصل المشي الكمي F_3 الى نقل كل الاحتمال الكلي الا بعد 6000 خطوة.



الوثيقة 05-3: تطور قيمة الاحتمال الممتص في بنية مثل الحلقة ذَات الثمانية مستويات بدلالة عدد الخطوات.

حيث نلاحظ أن الحلقة مكافئة تماما للخيط المغلق، من حيث سرعة انتقال الاحتمال. و من جهة أخرى يتضح أن انتقال الاحتمال يكون في حالة الاتجاهين أسرع من أن نضيف خطوة استراحة كاتجاه ثالث.

3) المشي الكمي G_3 يكون بالتقريب مساوٍ لسرعة المشي الكلاسيكي C_3 خلال الـ 25 خطوة الأولى، ثم تتناقص سرعته ليصبح هو الأقل سرعة من بين الجميع، و لكن لن يتمكن من نقل كل احتماله الكلي حيث لن يتجاوز نقل نسبة 45% من الاحتمال الكلي مهما زدنا في عدد الخطوات (وصلنا إلى مليون خطوة).

الفصل الرابع:

المشي العشوائي الكلاسيكي و المشي الكمي عبر بنى الجرافين

في هذا الفصل سنطبق فكرة المشي العشوائي الكلاسيكي و المشي الكمي على بنى الجرافين ذات البنية الكربونية (لن نراعي هنا المهية الفيزيائية و الكيميائية لذرات الكربون بل سنتعامل معها كعقد مجردة فقط)، و التي تتميز بتناظراتها و بنيتها المعقدة نوعا ما مقارنة بما تعرضنا إليه في الفصلين السابقين.

سنبدأ بجزيء الكربون C_{60}، و ذلك لتناظره الكروي المميز عموما، و تناظرات مضلعاته الخماسية و السداسية خصوصا، و التي سيكون لها دور مهم أثناء توظيفها في اختيار مجموعة العقد التي ستشكل موضع الانطلاق و مجموعة العقد التي ستشكل موضع الهدف. في الحقيقة لقد عملنا على اختبار و تحليل نتائج عدة مقادير، كتطور توزع الاحتمال، تطور أعلى قمة احتمال، تطور القيمة المتوسطة ...إلخ، لنصل في الأخير لاختبار خاصية الانتقال (الاحتمال الواصل).

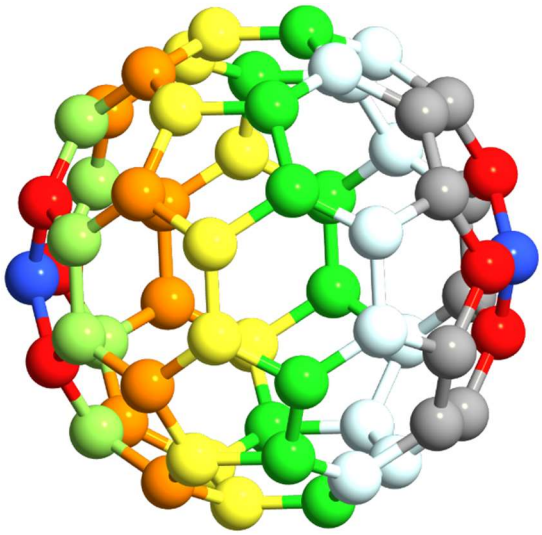
النتائج التي تحصلنا عليها هي التي شجعتنا للمضي قدما في دراسة بنى أخرى تتمتع ببعض التناظرات المميزة مثل بنى انابيب النانو بمختلف أصنافها، و التي تتميز بتناظرها الأسطواني عموم، و تنوع تراكيبها و واقعيتها المخبرية و الصناعية خصوصا. حيث نجد أنابيب النانو على شكل خاتم مغلق بصنفين: zigzag و armchair، و أنابيب مستقيمة مقببة بصنفين: zigzag و armchair، كلها أعطت نتائج مثيرة و أكثر كفاءة بخصوص مشيها الكمي مقارنة مع مشيها الكلاسيكي و حتى أنها أفضل من المشي الكمي لبعض البنى المشهورة و المدروسة سلفا كالحلقة و الخط.

1.4 جزيء الكربون C_{60}:

يتكون جزيء الكربون C_{60} من 60 ذرة كربون (أي $N=60$)، كل ذرة لها ثلاث وصلات (أي $d=3$) تتصل بثلاث ذرات مجاورة لها مشكلة بذلك بنية مغلقة على شكل كرة القدم التي نعرفها، حيث يظهر لنا 12 مضلع خماسي و 20 مضلع سداسي. سنختبر عبر بنية هذا الجزيء خاصية الانتقال سواء من خلال المشي الكلاسيكي أو المشي الكمي، في حالة ثلاثة اتجاهات ممكنة أو أربعة اتجاهات ممكنة إذا أضفنا خطوة الاستراحة. باستعمال فكرة المستويات (الوثيقة 04-2). سنتمكن من مقارنة نتائجنا مع الحلقة، ثم نتقصى أي الحالات هي الأكثر فعالية من بين الجميع في نقل الاحتمال الكلي. في عملية اختيارنا لموضع الانطلاق و موضع الهدف سنجرب ثلاث اختيارات و ذلك بحسب التناظرات المتوفرة في جزيء C_{60}. (**ملاحظة**: كل مجسمات بنى الجرافين المعروضة في هذا الفصل تم بناؤها باستعمال البرنامج Virtual Nanolab [62]).

1.1.4 C_{60} ذو المستويات العشرة:

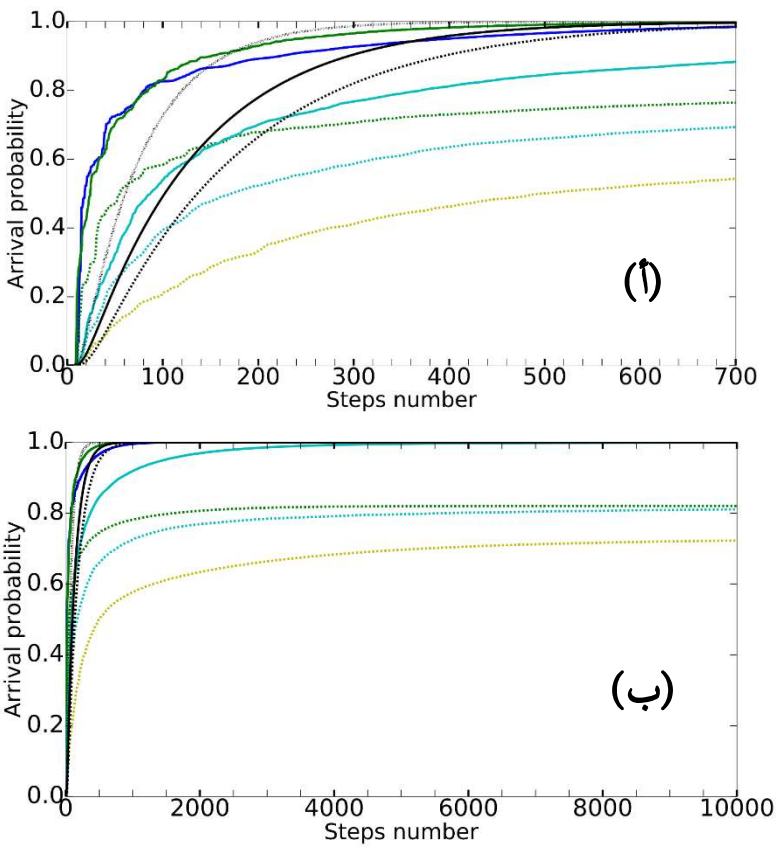
سنختار أحد العقد المنفردة كموضع انطلاق، و العقدة التي تناظرها قُطريا من الجهة الأخرى كموضع وصول أو هدف، و بالتالي سنحصل على بنية ذات 10 مستويات (الوثيقة 01-4). بعد تحرير البرنامج و تنفيذه (الملحق: البرنامج 3) و مقارنة النتائج مع نتائج الحلقة ذات الـ 10 مستويات C_{18} (أي حلقة بـ 18 عقدة)، سنجد النتائج المبينة في الوثيقة أ-02-4، أين يظهر جليا أن: (1) خلال الـ 190 خطوة الأولى تكون نتائج المشي الكمي أفضل من المشي الكلاسيكي (نقل أكثر من 90%).



الوثيقة 01-4: جزيء الكربون C_{60} ملون بطريقة لتحديد مختلف مستوياته العشر. انطلاقا من **موضع الانطلاق**: المستوى 0 (واحدة، زرقاء، يسارا). ثم المستوى 1 (ثلاثة، حمراء، يسارا). المستوى 2 (ستة، أخضر ليموني). المستوى 3 (ثمانية، برتقالي). المستوى 4 (عشرة، صفراء). المستوى 5 (عشرة، خضراء). المستوى 6 (عشرة، بيضاء). المستوى 7 (ثمانية، رمادي). المستوى 8 (ثلاثة، حمراء، يمينا). وصولا إلى **موضع الهدف**: المستوى 9 (واحدة، زرقاء، يمينا).

حيث تارة تكون الغلبة لمؤثر $\mathbb{G}_3$ بالنسبة لجزيء الكربون C_{60} و تارة أخرى لمؤثر $\mathbb{H}_2$ بالنسبة للحلقة C_{18}، إلى غاية الخطوة 90 أين تبقى الغلبة دائما لمؤثر $\mathbb{G}_3$ بالنسبة لجزيء الكربون C_{60}. (2) بعد الخطوة 190 يتفوق المشي الكلاسيكي (الحلقة C_{18}) على المشي الكمي حيث يصل هو أولا لنقل كامل احتماله الكلي قبل الجميع. (3) من الوثيقة ب-02-4 نلاحظ أن كل حالات المشي العشوائي تتمكن من نقل كل الاحتمال الكلي باستثناء الحالات الكمية التي تحتوي على خطوة الاستراحة ($\mathbb{G}_4$، $\mathbb{F}_4$ و $\mathbb{H}_4$).

تنبيه: من الآن فصاعدا، بالنسبة للمشي الكمي سنستعرض فقط نتائج مؤثر $\mathbb{G}_3$ بالنسبة لبنى الجرافين المتبقية التي سندرسها. و ذلك لأن بقية المؤثرات ستعطي نتائج أقل كفاءة في نقل الاحتمال الكلي مقارنة بالمؤثر $\mathbb{G}_3$.

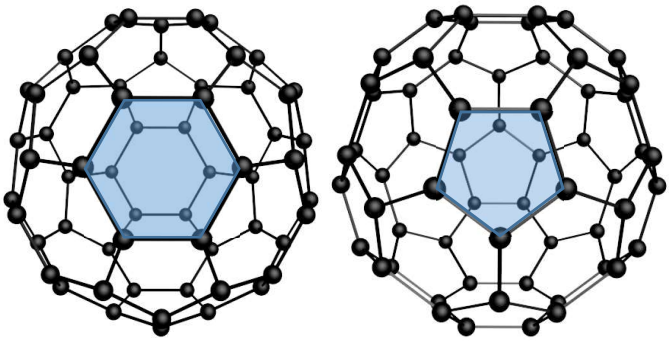


الوثيقة 02-4: التمثيل البياني لتزايد قيمة الاحتمال الواصل بدلالة عدد الخطوات. لاختبار خاصية الانتقال عبر جزيء الكربون C_{60} ذو العشر مستويات (الوثيقة 01-3). في حالة المشي الكمي: $\mathbb{G}_3$ (أخضر. مستمر). $\mathbb{G}_4$ (أخضر، متقطع). $\mathbb{F}_3$ (فيروزي. مستمر). $\mathbb{F}_4$ (فيروزي، متقطع). $\mathbb{H}_4$ (أصفر، متقطع). و المشي الكلاسيكي: ثلاثي الاتجاهات (أسود، مستمر). رباعي الاتجاهات (أسود، متقطع). للمقارنة أضفنا نتائج الحلقة ذات العشر مستويات C_{18}. في حالة المشي الكمي: $\mathbb{H}_2$ (أزرق. مستمر). و في حالة المشي الكلاسيكي: اتجاهين (أسود، منقط).

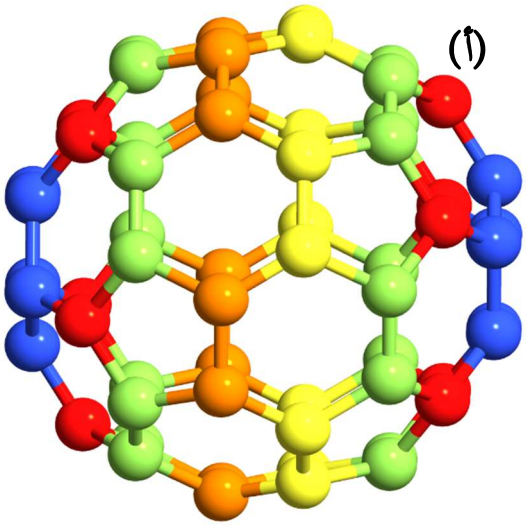
2.1.4 C_{60} ذو المستويات الثمانية:

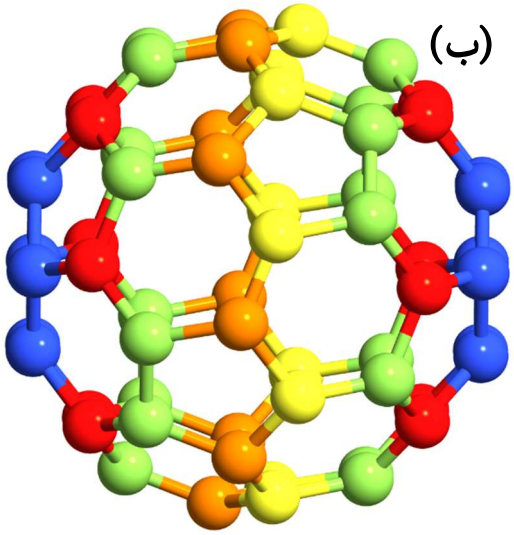
سنحاول توظيف التناظرات الموجودة في جزيء C_{60} و المتمثلة في المضلعات الخماسية و المضلعات السداسية (الوثيقة 03-4)، حيث سنختار عقد أحد المضلعات الخماسية (أي خمس عقد، على

شكل تركيبة خطية متساوية المعاملات) كموضع انطلاق و عقد المضلع الخماسي الذي يناضره قطريا في الجهة المقابلة كموضع وصول. و بنفس الطريقة سنختار كذلك عقد أحد المضلعات السداسية (أي ست عقد، على شكل تركيبة خطية متساوية المعاملات) كموضع انطلاق و عقد المضلع السداسي الذي يناظره قطريا في الجهة المقابلة كموضع وصول. و بالتالي سنحصل على بنية ذات 8 مستويات في الحالتين (الوثيقة 04-4).



الوثيقة 03-4: استعراض بنية جزيء الكربون C_{60} لتوضيح كيف أن حالة استعمال المضلع السداسي (يسارا) كزوج: انطلاق/هدف. يكون أكثر تناظرا من استعمال المضلع الخماسي (يمينا) كزوج: انطلاق/هدف.



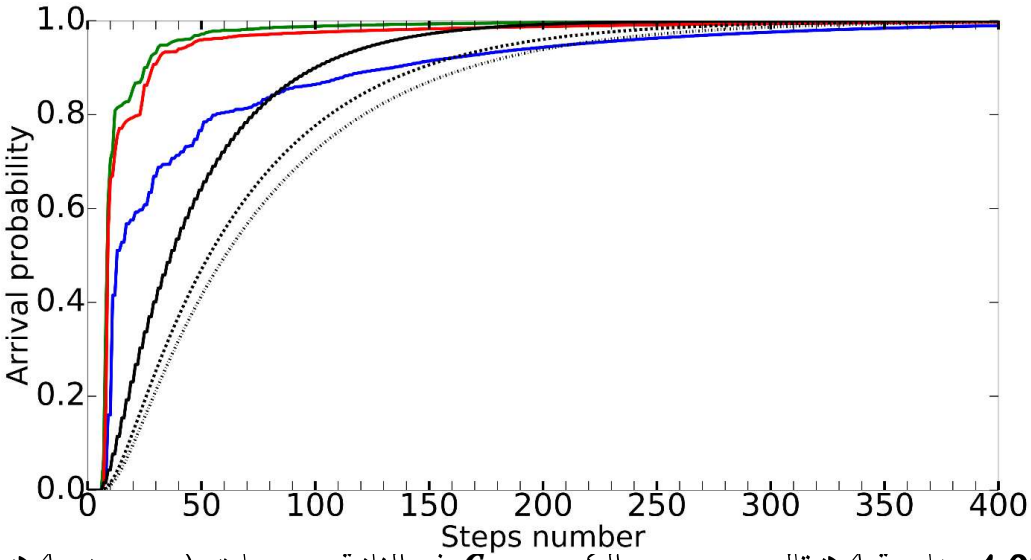


الوثيقة 04-4: جزيء الكربون C_{60} ملون بطريقة لتحديد مختلف مستوياته **الثمانية**.

(أ) حالة المضلع الخماسي: انطلاقا من **موضع الانطلاق**: المستوى 0 (خمسة على شكل خماسي، زرقاء، يسارا). ثم المستوى 1 (خمسة، حمراء، يسارا). المستوى 2 (عشرة، ليموني، يسارا). المستوى 3 (عشرة، برتقالي). المستوى 4 (عشرة، صفراء). المستوى 5 (عشرة، ليموني، يمينا). ب. المستوى 6 (خمسة، حمراء، يمينا). وصولا إلى **موضع الهدف**: المستوى 7 (خمسة على شكل خماسي، زرقاء، يمينا).

(ب) حالة المضلع السداسي: انطلاقا من **موضع الانطلاق**: المستوى 0 (ستة على شكل سداسي، زرقاء، يسارا). ثم المستوى 1 (ستة، حمراء، يسارا). المستوى 2 (تسعة، ليموني، يسارا). المستوى 3 (تسعة، برتقالي). المستوى 4 (تسعة، صفراء). المستوى 5 (تسعة، ليموني، يمينا). ب. المستوى 6 (ستة، حمراء، يمينا). وصولا إلى **موضع الهدف**: المستوى 7 (ستة على شكل سداسي، زرقاء، يمينا).

بعد تحرير البرنامج و تنفيذه (الملحق: البرنامج 4 و 5) و مقارنة النتائج مع نتائج الحلقة ذات الـ 8 مستويات C_{14} (أي حلقة بـ 14 عقدة)، سنجد النتائج المبينة في الوثيقة 05-4، و التي تظهر جليا هذه المرة أن: (1) المشي الكمي يبقى دائما هو الأسرع من المشي الكلاسيكي و يصل أولاً الى نقل كل احتماله الكلي خلاف اختيار حالة العقدة المنفردة. (2) كلما زدنا من درجة التناظر في اختيارنا لعقد موضع الانطلاق/الهدف في المشي الكمي كلما زادت كفاءة نقل الاحتمال الكلي في جزيء C_{60}، حيث أن اختيار المضلع السداسي يكون أكفأ من اختيار المضلع الخماسي و كلاهما دائما أكفأ من الحلقة و بفارق ملموس. عكس حالة اختيار العقدة المنفردة في جزيء C_{60} أين تكون الغلبة تارة له و تارة أخرى للحلقة.

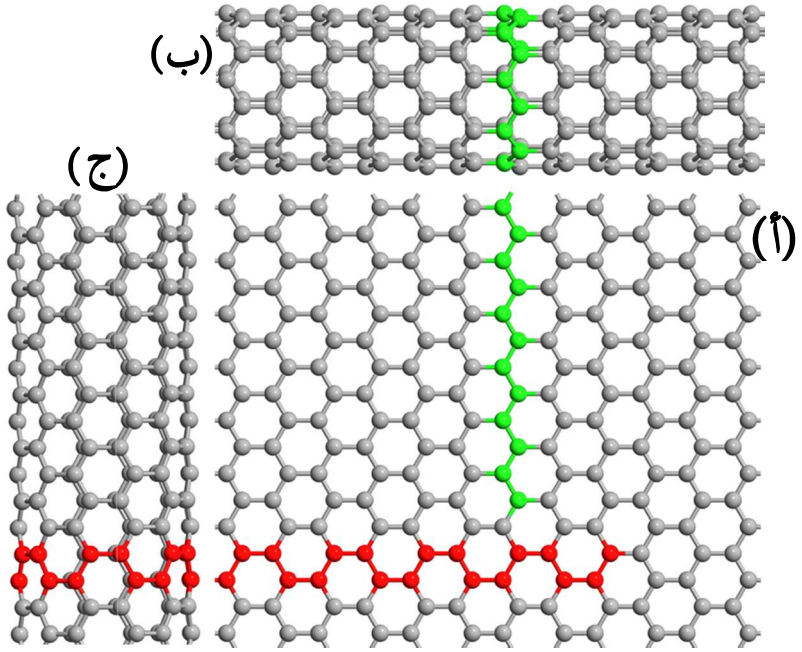


الوثيقة 05-4: خاصية الانتقال عبر جزيء الكربون C_{60} ذو الثمانية مستويات (بين موضع الانطلاق (المستوى 0) و موضع الهدف (المستوى 7) (الوثيقة 01-4)). التمثيل البياني لتزايد قيمة الاحتمال الواصل بدلالة عدد الخطوات (400 خطوة)، في حالة المشي الكمي باستخدام $\mathbb{G}_3$ و لمختلف وضعيات الانطلاق: سداسي (أخضر. مستمر). الخماسي (أحمر، مستمر). و المشي الكلاسيكي: السداسي (أسود، متقطع). الخماسي (أسود، منقط). للمقارنة أضفنا نتائج الحلقة ذات الثماني مستويات C_{14}. في حالة المشي الكمي: $\mathbb{H}_2$ (أزرق. مستمر). و في حالة المشي الكلاسيكي: ثنائي الاتجاهات (أسود، مستمر).

2.4 أنابيب النانو الكربونية (الـ Zig-Zag و الـ Arm-chair):

كما هو معلوم فإن بنى أوراق الجرافين كلها تتكون من ذرات الكربون ذات ثلاث وصلات (أي $d=3$)، تتصل بثلاث ذرات مجاورة لها مشكلة سداسيات متتالية و متراصة في شكل منبسط و مستو. إذا لفت وفق اتجاه الـ zigzag فإننا سنتحصل على أنبوب مفتوح من نوع zigzag. و إذا

لفت وفق اتجاه الـ armchair فإننا سنتحصل على أنبوب مفتوح من نوع armchair (الوثيقة 06-4). الانابيب المفتوحة هي بنى غير منتظمة و ذلك بسبب العقد الطرفية التي تملك وصلتين فقط (d=2)، و لحل هذه المشكلة اهتدينا لطريقتين:



الوثيقة 06-4: (أ) ورقة جرافين على شكل سداسيات متراصة. حيث نميز فيها اتجاهين: zigzag (أخضر). armchair (أحمر)
(ب) إذا لفت وفق اتجاه zigzag نحصل على أنبوب من نوع zigzag.
(ج) إذا لفت وفق اتجاه armchair نحصل على أنبوب من نوع armchair.

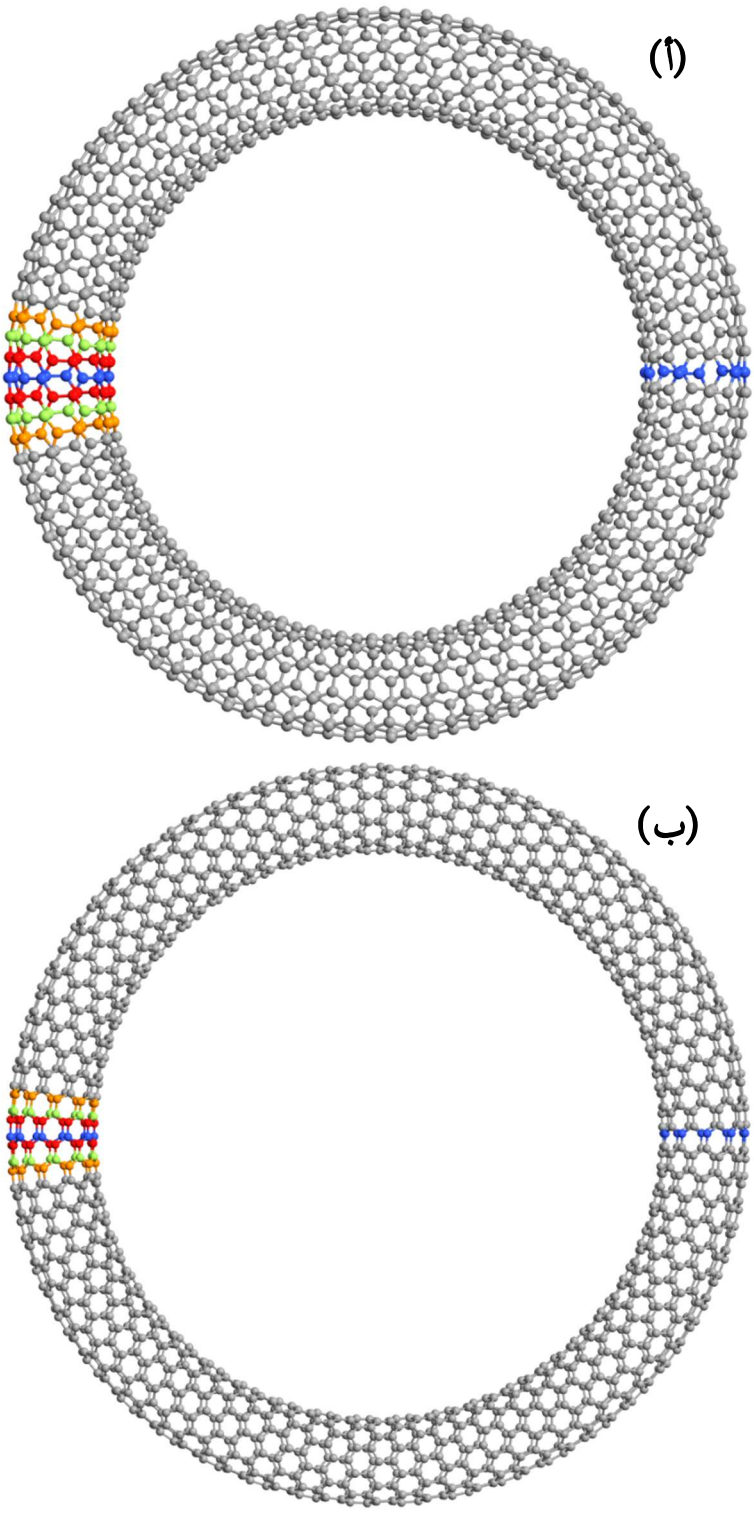
1.2.4 الأنابيب المغلقة على شكل حلقة (أو الحلقية):

و هي وصل الأطراف المفتوحة لأنابيب النانو مع بعضها البعض، لتشكيل أنبوب مغلق على شكل حلقة أو لنقل خاتم دائري الشكل، و بالتالي كل عقدة ستكون لها ثلاث وصلات (الوثيقة 07-4). عمليا هذه الطريقة تماثل تماما فكرة ربط الطرفين المفتوحين للخط المفتوح لتشكيل حلقة. لدراسة خاصية الانتقال عمدنا لتوظيف التناظر مرة أخرى في اختيار موضع الانطلاق/ الهدف. حيث استعملنا مجموعة العقد التي تقع على نفس المستوى (مشكلة دائرة عمودية على المحور الطولي للأنبوب) كموضع انطلاق و مجموعة العقد التي تناظرها قطريا و تقع على نفس المستوى (أو الدائرة) موضع هدف (الوثيقة 07-4). كما قمنا بتجريب تغيير نصف قطر الأنبوب و طوله كل مرة لمعرفة مدى تأثيرهما على خاصية الانتقال.

1.1.2.4 المقارنة بين أنابيب الـ Zig-Zag و الـ Arm-chair:

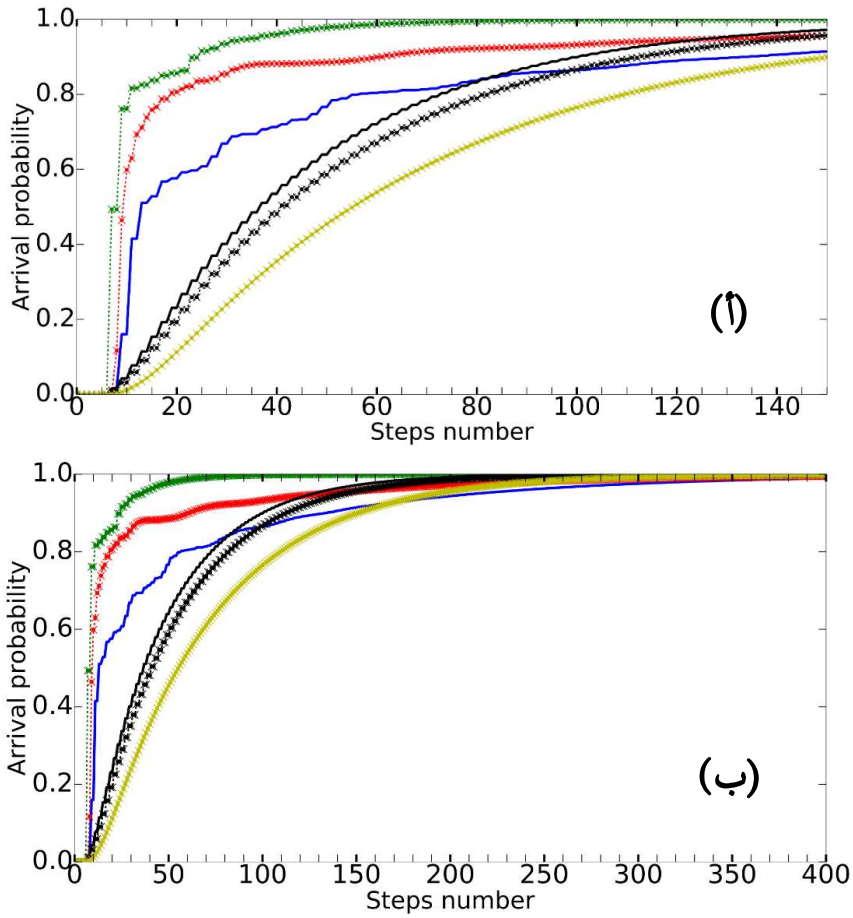
بعد تحرير البرنامج و تنفيذه (الملحق: البرنامج 6 و 7). تحصلنا على النتائج الموضحة في الوثيقة 08-4. و التي تظهر أن: (1) مهما غيرنا نصف قطر الانابيب فستبقى سرعة الانتقال هي نفسها سواء بالنسبة للمشي الكمي أو الكلاسيكي، و بالتالي خاصية الانتقال لا تتعلق بنصف قطر الانبوب

(الوثيقة أ-08-4). (2) المشي الكمي في أنابيب النانو يكون هو الأفضل دائما، خاصة بالنسبة انابيب الـ zigzag التي تصل أولا لنقل كل الاحتمال مقارنة بالجميع. (3) تفوق انابيب الـ zigzag على أنابيب armchair (الوثيقة ب-08-4). (4) كلما زدنا في طول الانابيب كلما قلت كفاءة الانتقال و لكن مع بقاء النتائج السابقة صحيحة.

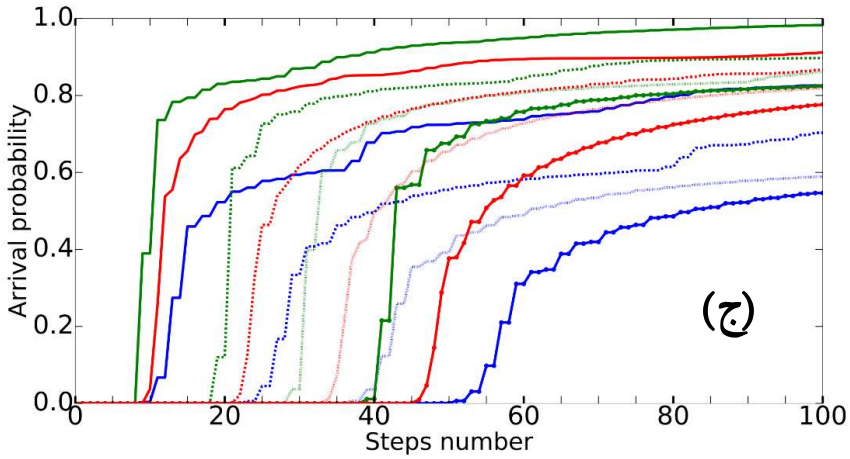


الوثيقة 07-4: أنابيب النانو الكربونية المغلقة على شكل خاتم، و كيفية اختيارنا لموضع الانطلاق من المستوى 0 (مجموعة العقد باللون الأزرق جهة اليسار) الذي ينطلق منه المشي الكمي و يتقدم تدريجيا عبر مختلف المستويات: المستوى 1 (مجموعة العقد باللون الأحمر)، المستوى 2 (مجموعة العقد باللون الليموني)، المستوى 3 (مجموعة العقد باللون البرتقالي)، هكذا و تواليك...إلى غاية الوصول إلى موضع الهدف (مجموعة العقد باللون الأزرق جهة اليمين). **(أ)** انابيب من نوع arm-chair مشكلة 56 مستوى. **(ب)** انابيب من نوع zig-zag مشكلة 91 مستوى.

الوثيقة 08-4:



(أ)+(ب): خاصية الانتقال عبر أنابيب النانو المغلقة كالخاتم (ذات 8 مستويات) و بعدة اقطار مختلفة: 6 عقد (×)، 10 عقد (متقطع) و 14 عقدة (•). في حالة المشي الكمي نجد أنابيب zigzag (أخضر) و أنابيب armchair (أحمر). في حالة المشي الكلاسيكي نجد أنابيب zigzag (أسود) و أنابيب armchair (أصفر). للمقارنة أضفنا نتائج الحلقة ذات ثمانية مستويات C_{14}. في حالة المشي الكمي: $\mathbb{H}_2$ (أزرق. مستمر). و في حالة المشي الكلاسيكي: ثلاثي الاتجاهات (أسود، مستمر). **(أ)** في حالة 150 خطوة. **(ب)** في حالة 400 خطوة.

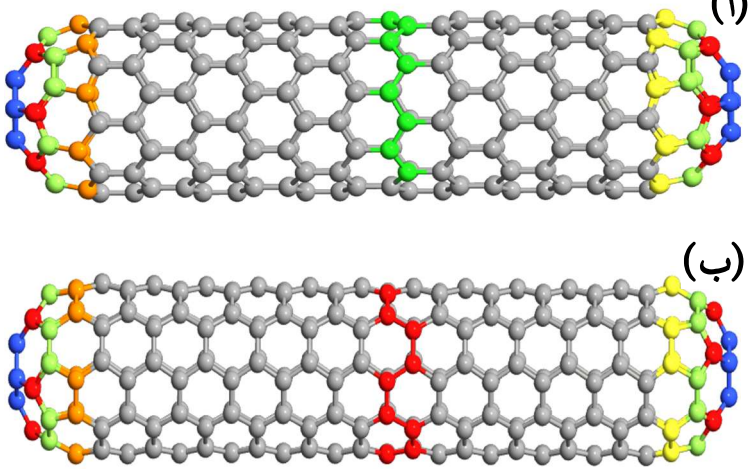


(ج) خاصية الانتقال لمشي الكمي عبر أنابيب النانو المقببة: أنابيب zigzag (أخضر). أنابيب armchair (أحمر) و الحلقة (أزرق). و لكن لأطوال مستويات مختلفة: 10 (مستمر)، 20 (متقطع)، 30 (منقط) و 40 (مستمر، •).

2.2.4 الأنابيب المغلقة على شكل كبسولة (أو المقببة):

و هي وصل الطرفين المفتوحين لأنابيب النانو بالنصفين الكرويين لجزيء C_{60} على الترتيب، حيث

بحسب **التكامل البنيوي** يمكننا فقط: **(أ)** وصل أنابيب الـ zigzag ذات القطر الثابت الذي يحتوي على 9 عقد، مع النصفين الكرويين الموضحين في الوثيقة ب-04-4 و اللذان يحتويان على 9 عقد بالضبط عند طرفيها (أي عند منتصف كرة C_{60} في المستوى الثالث (اللون البرتقالي) و الرابع (اللون الأصفر))، و بالتالي سيكون موضع الانطلاق/الهدف هما عقد المضلعين السداسيين المعلمين باللون الأزرق في الوثيقة أ-09-4. **(ب)** وصل أنابيب الـ armchair ذات القطر الثابت الذي يحتوي على 10 عقد، مع النصفين الكرويين الموضحين في الوثيقة أ-04-4 و اللذان يحتويان على 10 عقد بالضبط عند طرفيها (أي عند منتصف كرة C_{60} في المستوى الثالث (اللون البرتقالي) و الرابع (اللون الأصفر)). و بالتالي سيكون موضع الانطلاق/الهدف هما عقد المضلعين الخماسيين المعلمين باللون الأزرق في الوثيقة ب-09-4، و بالتالي الحصول على بنية منتظمة كل عقدها لها نفس عدد الوصلات (d=3).



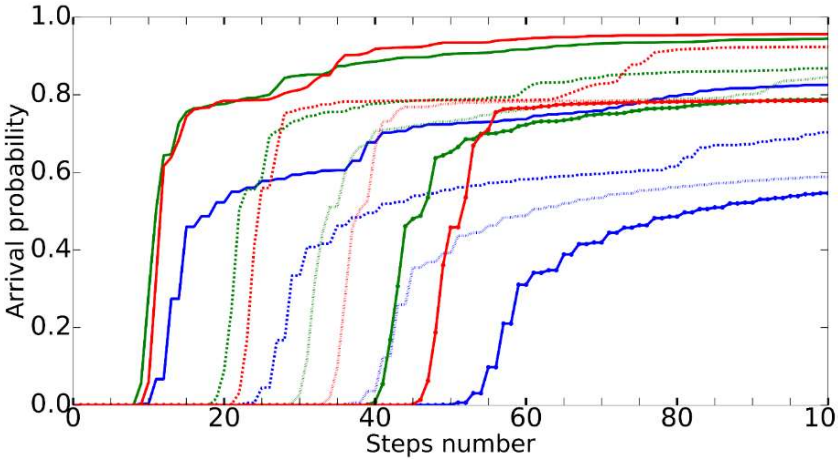
الوثيقة 09-4: أنابيب النانو المقببة **(أ)** ذات 30 مستوى. انطلاقا من المستوى الصفري (العقد الزرقاء يسارا: (أ) 6 عقد بالنسبة لأنابيب zigzag. (ب) 5 عقد بالنسبة لأنابيب armchair) الى غاية الوصول إلى الهدف في المستوى 29 (العقد الزرقاء يمينا: (أ) 6 عقد بالنسبة لأنابيب zigzag. (ب) 5 عقد بالنسبة لأنابيب armchair). **(ب)**

من التكامل البنيوي الذي اعتمدنا عليه في وصل الانابيب النانو المفتوحة، نجد انفسنا مقيدين في إختباراتنا لهذه الانابيب، حيث لم يبقى لنا سوى اختبار خاصية طول الانابيب و علاقتها مع خاصية الانتقال.

1.1.2.4 المقارنة بين أنابيب الـ Zig-Zag و الـ Arm-chair:

بعد تحرير البرنامج و تنفيذه (الملحق: البرنامج: 6 و 7)، تحصلنا على النتائج الموضحة في الوثيقة 10-4، و التي تظهر أن: (1) الغلبة تكون متبادلة بين أنابيب الـ zigzag و أنابيب armchair، أحيانا لهذه و أحيانا لتلك، و لكن كلاهما يكون أسرع من نتائج الحلقة. (2) كلما زدنا في طول الانابيب

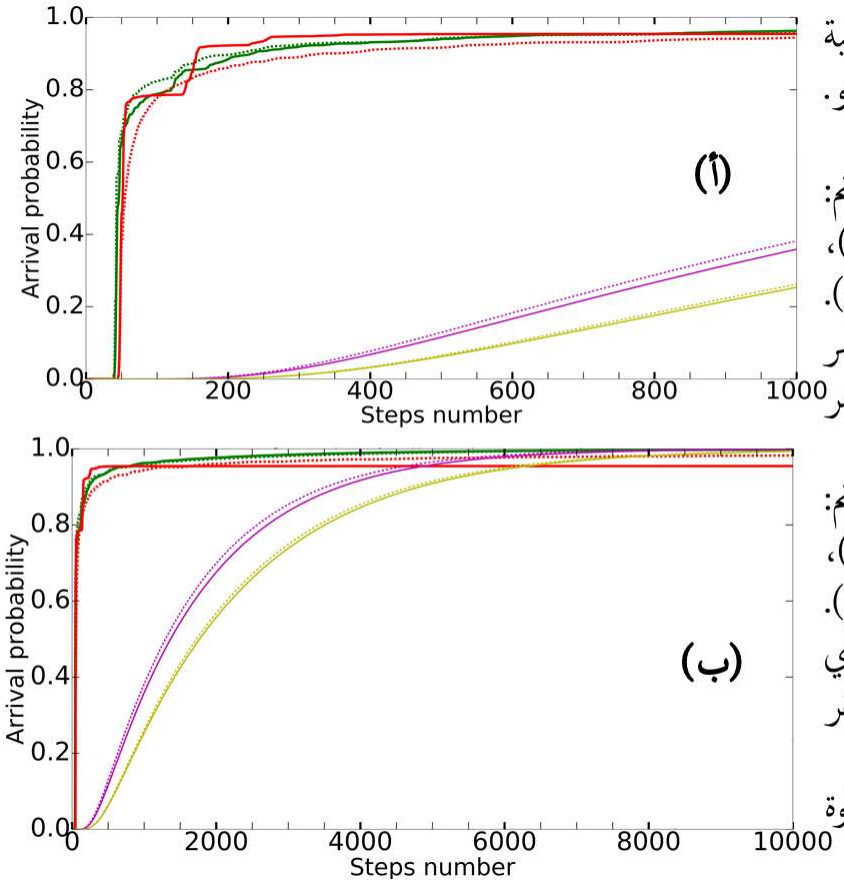
كلما قلّت كفاءة الانتقال.



الوثيقة 4-10: خاصية الانتقال لمشي الكمي عبر أنابيب النانو المقببة: zigzag (أخضر). armchair (أحمر) و الحلقة (أزرق). و لكن لأطوال مستويات مختلفة: 10 (مستمر)، 20 (متقطع)، 30 (منقط) و 40 (مستمر+ •).

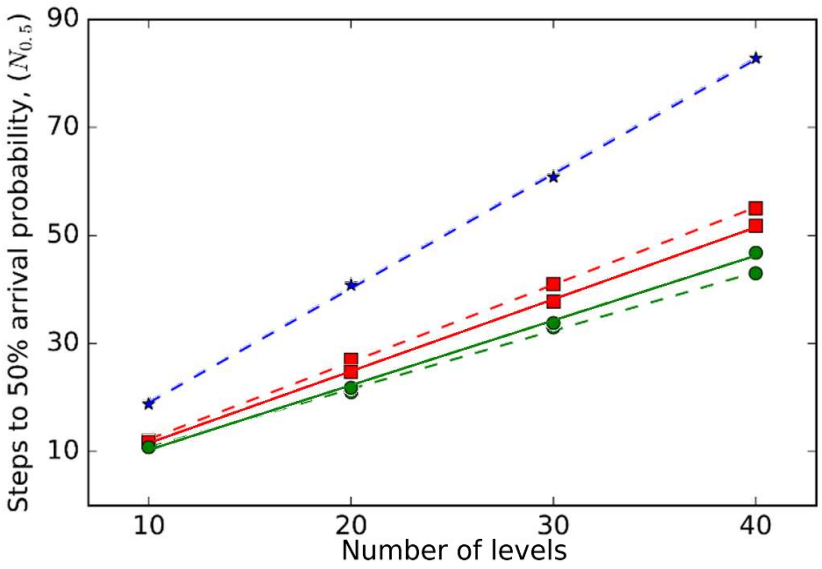
3.2.4 المقارنة بين مختلف أنابيب النانو المغلقة (الحلقية و المقببة):

عند مقارنتنا بين انابيب النانو المغلقة على شكل حلقة و المقببة في حالة طول معين ثابت (40 مستوى مثلا) نجد النتائج المبينة في الوثيقة 4-11، حيث أن الغلبة تكون في البداية لأنابيب الـ zigzag سواء المغلقة كالحلقة أو المقببة (نقل أكثر من 88% من الاحتمال الكلي) ثم تأتي فترة معينة (من 150 إلى 550 خطوة) أين يتفوق أنبوب الـ armchair المقبب و لكن بعدها سيثبت عند



الوثيقة 4-11: المقارنة لخاصية الانتقال بين الانابيب النانو. طول كل منها 40 مستوى.
المشي الكمي. المغلقة كالخاتم: zigzag (أخضر متقطع)، armchair (أحمر متقطع). المقببة: zigzag (أخضر مستمر)، armchair (أحمر مستمر).
المشي الكلاسيكي. المغلقة كالخاتم: zigzag (الوردي متقطع)، armchair (أصفر متقطع). المقببة: zigzag (الوردي مستمر)، armchair (أصفر مستمر).
(أ)،(ب): 1000، 10000 خطوة على الترتيب.

قيمة معينة (95%) و لا يزيد عليها مهما زدنا عدد الخطوات (60000 خطوة). فيما يخص الـ armchair المغلق كالخاتم يعتبر هو الابطئ من بين جميع الحالات الكمية و لكنه يصل إلى نقل جميع احتماله الكلي مع الوقت (30000 خطوة).



الوثيقة 12-4: تطور عدد الخطوات اللازمة لانتقال نصف الاحتمال الكلي ($N_{0.5}$) بدلالة طول أنابيب النانو. zigzag مغلقة (خط أخضر متقطع)، zigzag مقبب (خط أخضر مستمر)، armchair مقبب (خط أحمر مستمر)، armchair مغلق (خط أحمر متقطع). للمقارنة أضفنا الحلقة (خط أزرق متقطع).

الآن سنقدم دراسة معيارية نتقصى من خلالها العلاقة الرياضياتية لتناقص كفاءة خاصية الانتقال مع تزايد طول الأنابيب. سنختبر كم عدد الخطوات المطلوبة التي من أجلها يكون قد تم نقل أكثر من نصف الاحتمال الكلي (سنرمز لهذه القيمة بـ: $N_{0.5}$)، و ذلك بدلالة طول الأنابيب المختلفة. الوثيقة 12-4 توضح النتائج التي تحصلنا عليها، حيث بيانيا نلاحظ جليا أن نتائج كل أنبوب تتطور خطيا مع تغير طول الانبوب، و أن الغلبة في الكفاءة تكون لأنابيب الـ zigzag المغلقة كخاتم ثم المقببة، و بعدها أنابيب الـ armchair المقببة ثم المغلقة كالخاتم، و تأتي في الأخير نتائج الحلقة. كميّا، يمكننا التعبير عن كيفية تطور قيم كل أنبوب عن طريق معادلة خط مستقيم عبارته العامة من الشكل: $y = m\,x + b$، و بالتالي ستكون نتائج كل بنية ملخصة على النحو التالي:

نوعية البنية	m	b	r^2
الحلقة	2.12	-2.00	0.9996
الـ zigzag المغلقة	1.08	0.00	0.9986
الـ zigzag المقببة	1.20	-1.50	0.9986
الـ armchair المقببة	1.33	-1.50	0.9996
الـ armchair المغلقة	1.43	-2.00	0.9997

حيث r^2 هو ثابت يصف دقة هذا التعبير و يسمى بـ: coeffecient of determination.

الفصل الخامس:

تلخيص و مناقشة النتائج

في هذا الجهد المتواضع الذي كلل و لله الحمد، بنشره كمقال في مجلة محكمة [63]، نكون قد قدمنا برهانا نظريا عن طريق معالجة رقمية تبين أن خاصية الانتقال عن طريق المشي الكمي غير المستمر، تكون أكثر كفاءة و سرعة في بنى مختلفة من بنى الجرافين الواقعية مقارنة مع نظيراتها من نفس الحجم كالحلقة التي كانت في وقت ليس ببعيد تعد هي الأكفأ و الأسرع.

إن حسن توظيف تناظرات بنى الجرافين كجزيء الكربون C_{60} ذو التناظر الكروي و أنابيب النانو ذات التناظر الأسطواني، هو الذي كان له الدور المهم و المساعد في الوصول إلى هذه النتائج.

حسب المشي الكمي وجدنا أن الانتقال يكون أحسن في أنابيب الـ zigzag مقارنة بأنابيب الـ armchair، و هذا ما يخالف خواص الناقلية الكهربائية لأنابيب النانو، حيث أن أنابيب الـ zigzag لها سلوك أنصاف النواقل و أنابيب الـ armchair لها سلوك المعادن. و بالتالي نتائج هذا العمل من شأنها أن تؤدي إلى طرح أسئلة جديدة لمعرفة العلاقة بين المشي الكمي من جهة و الناقلية الكهربائية من جهة أخرى. و هذا كله سيعزز أكثر في فهم سلوكيات بنى النانو و التحكم فيها أكثر.

غالبا ما يكون سلوكا المشي الكمي الغير مستمر يماثل المشي الكمي المستمر [64]. و بالتالي من الأفضل بمكان التأكد من صلاحية نتائجنا حتى بالنسبة للمشي الكمي المستمر، و هذا ما سيخدم أيضا بعض المواضيع التي تخص النانو تكنولوجي أو بنى النانو في الوقت المستمر، و هي كثيرة و متعددة.

تعد نتائج هذه الدراسة عملية لتجنبها مشكلة "زمن الوصول غير المنتهي" حيث أن جميع بنى الجرافين المدروسة تتمكن من نقل جميع احتمالها الكلي للهدف باستثناء حالة واحدة فقط، و هي أنابيب الـ

armchair المقببة، و بالتالي هذا ما سيعزز من أهمية هذه الدراسة و قابلية تطبيق نتائجها.

بما أن الانتقال أو نقل الاحتمال الكلي عبر الحلقة يعد أسرع و أكفأ مقارنة بالخط من نفس الحجم [20]، إذن عمليا يمكننا القول أن بنى الجرافين هي الأخرى أكفأ و أسرع من الخط في نقل الاحتمال الكلي، و بالتالي ستخدم فكرة توظيف أنابيب النانو كأسلاك توصيل في الدارات الكمية أو حتى كمكون رئيس في المعالجات الكمية [19، 20].

الملحق:

جدول ترجمة المصطلحات التقنية:

المصطلحات العربية	المصطلحات الإنجليزية
المشي العشوائي الكلاسيكي	Classical random walk
المشي الكمي	Quantum walk
مخطاط	Graph
خاصية الانتقال	Transport property
شبكة خلية النحل	Honey comb lattice
عقد	Nodes, vertexes
وصلة	Edge
منتظم	regular
غير موجه	Undirected
عملة	coin
زمن الاصطدام	Hitting time
زمن الاستقرار أو زمن التمازج	Mixing time
التوزيع النهائي	Limit distribution
مؤثر القفزات	Shift operator
مؤثر الشقلبة	Flip-flop operator
مؤثر العملة	Coin operator
ورقة الجرافين (أو الغرافين)	Graphene sheet
مضلع خماسي	Hexagonal
مضلع سداسي	Pentagonal
انابيب النانو	Nanotube
منعرج	Zigzag
ذراع الكرسي	armchair
أنبوب مقبب	Capped tube
أنبوب حلقي	Torus tube

البرامج الحاسوبية:

البرنامج 1: تطور توزع الاحتمال الكلي للخط المفتوح و الحلقة

```
print ('بسم الله الرحمان الرحيم')
#Author: Hamza Bougroura
#------ Import the requirement module ---------------------------------------------
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
from matplotlib.ticker import MultipleLocator
#------ Define the critical number ------------------------------------------------
T = 200         # number of random steps
N = (2*T)+1     # number of node or vertex in open line  (in cycle case we fix the value of N (for example N= 11)
O=1             # O,X,Y to control the different initial state (the coin part)
X=1
Y=1
#===== operators ====================================
G3 = (np.array([[-1, 2, 2], [2, -1, 2], [2, 2, -1]]))/(3.)
F3 = (np.fft.fft(np.eye(3)))/np.sqrt(3.)
H2 = (np.array([[1, 1], [1, -1]]))/np.sqrt(2)
#=====coin state ====================================
coin2 = np.array([[1, 0], [0, 1]])
coin3 = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
#---------------
cxx2 = np.outer(coin2[0], coin2[0])
cyy2 = np.outer(coin2[1], coin2[1])
#--------------
coo3 = np.outer(coin3[0], coin3[0])
cxx3 = np.outer(coin3[1], coin3[1])
cyy3 = np.outer(coin3[2], coin3[2])
#====== Define steps ================================
node = np.eye(N)
k=0
step_plus=0
while k<=(N-2):
        plus = np.outer(node [k+1], node [k])
        step_plus = step_plus + plus
        k+=1
step_plus = step_plus + np.outer(node [0], node [N-1])
#========
k=1
step_minus=0
while k<=(N-1):
        minus = np.outer(node [k-1], node [k])
        step_minus = step_minus + minus
        k+=1
step_minus = step_minus + np.outer(node [N-1], node [0])
#========
k=0
step_rest=0
while k<=(N-1):
        rest = np.outer(node [k], node [k])
        step_rest = step_rest + rest
        k+=1
#===== compute the Shift operator ======================
S_Q2 = np.kron(step_plus, cxx2) + np.kron(step_minus, cyy2)
S_C2 = step_plus + step_minus
S_Q3 = np.kron(step_rest, coo3) + np.kron(step_plus, cxx3) + np.kron(step_minus, cyy3)
S_C3 = step_rest + step_plus + step_minus
#===== Evolution operator ======================
UH2 = S_Q2.dot(np.kron(np.eye(N), H2))
UG3 = S_Q3.dot(np.kron(np.eye(N), G3))
UF3 = S_Q3.dot(np.kron(np.eye(N), F3))
UC2 = S_C2/(2.)
UC3 = S_C3/(3.)
#==== Define the initial state Psi(zero) =============
posn0 = np.zeros(N)
posn0[(N-1)/2] = 1 # the initial node position space
psi0Q2 = np.kron(posn0,((coin2[0])*X+(coin2[1])*Y)/np.sqrt(X+Y))
psi0C2 = posn0
psi0Q3 = np.kron(posn0,((coin3[0]*O)+(coin3[1]*X)+(coin3[2]*Y))/np.sqrt(X+Y+O))
psi0C3 = posn0
#==========evolution equation=======================
psiNH2 = np.linalg.matrix_power(UH2, T).dot(psi0Q2)
psiNC2 = np.linalg.matrix_power(UC2, T).dot(psi0C2)
psiNG3 = np.linalg.matrix_power(UG3, T).dot(psi0Q3)
```

```
psiNF3 = np.linalg.matrix_power(UF3, T).dot(psi0Q3)
psiNC3 = np.linalg.matrix_power(UC3, T).dot(psi0C3)
#======= compute the Probability========
#=====Hadamard: 2D=================
probH2 = np.zeros(N)
k = 0
while k<N:
        posn = np.zeros(N)
        posn[k] = 1
        M_hat_k = np.kron( np.outer(posn,posn), np.eye(2))
        proj = M_hat_k.dot(psiNH2)
        probH2[k] = proj.dot(proj.conjugate()).real
        k+=1
xH2=np.sum(probH2)
#======Grover: 3D===================
probG3 = np.zeros(N)
k = 0
while k<N:
        posn = np.zeros(N)
        posn[k] = 1
        M_hat_k = np.kron( np.outer(posn,posn), np.eye(3))
        proj = M_hat_k.dot(psiNG3)
        probG3[k] = proj.dot(proj.conjugate()).real
        k+=1
xG3=np.sum(probG3)
#======Fourier: 3D====================
probF3 = np.zeros(N)
k = 0
while k<N:
        posn = np.zeros(N)
        posn[k] = 1
        M_hat_k = np.kron( np.outer(posn,posn), np.eye(3))
        proj = M_hat_k.dot(psiNF3)
        probF3[k] = proj.dot(proj.conjugate()).real
        k+=1
xF3=np.sum(probF3)
#======classic: 2D ======================
xC2=np.sum(psiNC2)
probC2=psiNC2
#======classic: 3D ======================
xC3=np.sum(psiNC3)
probC3=psiNC3
#======Plot the probability====================
minorLocator = AutoMinorLocator()
fig, ax = plt.subplots()
plt.plot(probH2, 'r*-', label='$H_2$', markersize=20, linewidth=7)
plt.plot(probC2, '*-', label='$C_2$', markersize=25, linewidth=7, c='gray')
plt.plot(probG3, 'go-', label='$G_3$', markersize=10, linewidth=7, c='lime')
plt.plot(probF3, 'bo-', label='$F_3$', markersize=10, linewidth=7)
plt.plot(probC3, 'ko--', label='$C_3$', markersize=10, linewidth=7)
V = N-1
plt.xticks(range (0, N, (V)/10))
ax.set_xticklabels(range (-V/2, (V/2)+1, (V)/10))
A = np.array([[probC2], [probC3], [probF3], [probG3], [probH2]])
y = (np.max (A))-0.05
plt.text(((N-1)/2), y, '$T= '+str(T)+'$', fontweight='heavy',bbox={'facecolor':'red', 'alpha':0.5, 'pad':10}, fontsize=65)
ax.xaxis.set_minor_locator(minorLocator)
ax.yaxis.set_minor_locator(minorLocator)
ax.grid(which='both')
ax.grid(which='minor', lw=0.5, ls=':')
ax.grid(which='major', lw=1, ls='--')
ax.tick_params(axis='x', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='x', which='minor', labelsize=70, width=2, length=10, colors='k')
ax.tick_params(axis='y', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='y', which='minor', labelsize=70, width=2, length=10, colors='k')
ax.set_xlabel('Steps number', fontsize=65)
ax.set_ylabel('Probability', color='k', fontsize=65)
for tl in ax.get_yticklabels():
tl.set_color('k')
ax.xaxis.set_minor_locator(MultipleLocator(50))
ax.yaxis.set_minor_locator(MultipleLocator(0.10))
plt.title('quantum walk in simple line aftre ' + str(N) + ' steps', fontsize=16)
plt.legend(fontsize=50)
figManager = plt.get_current_fig_manager()
figManager.window.showMaximized()
plt.tight_layout()
plt.show()
print ('الحمد لله')
#=== End of the code=========
```

البرنامج 2: تطور الاحتمال الواصل للحلقة

```
print ('بسم الله الرجمان الرحيم')
#Author: Hamza Bougroura
#------ Import the requirement module ----------------------
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
from matplotlib.ticker import MultipleLocator
#------ Define the critical number -------------------------
T = 500          # number of random steps
a = 2            # just a constant
N = (a*6)+a      # number of node (we change only the value of 6)
Lev = (N/2)+1    # level (circle of 8 level)
O=1
X=1
Y=1
#== operators ================================================
G3 = (np.array([[-1, 2, 2], [2, -1, 2], [2, 2, -1]]))/(3.)
F3 = (np.fft.fft(np.eye(3)))/np.sqrt(3.)
H2 = (np.array([[1, 1], [1, -1]]))/np.sqrt(2)
#=====coin state ==============================================
coin2 = np.array([[1, 0], [0, 1]])
coin3 = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
#------------
cxx2 = np.outer(coin2[0], coin2[0])
cyy2 = np.outer(coin2[1], coin2[1])
#--------------
coo3 = np.outer(coin3[0], coin3[0])
cxx3 = np.outer(coin3[1], coin3[1])
cyy3 = np.outer(coin3[2], coin3[2])
#======== Define different steps (the matrix) ================
node = np.eye(N)
k=0
step_plus=0
while k<=(N-2):
        plus = np.outer(node [k+1], node [k])
        step_plus = step_plus + plus
        k+=1
step_plus = step_plus + np.outer(node [0], node [N-1])
#----------
k=1
step_minus=0
while k<=(N-1):
        minus = np.outer(node [k-1], node [k])
        step_minus = step_minus + minus
        k+=1
step_minus = step_minus + np.outer(node [N-1], node [0])
#----------
k=0
step_rest=0
while k<=(N-1):
        rest = np.outer(node [k], node [k])
        step_rest = step_rest + rest
        k+=1
#================= the Shift operator ============================
S_Q2 = np.kron(step_plus, cxx2) + np.kron(step_minus, cyy2)
S_C2 = step_plus + step_minus
S_Q3 = np.kron(step_rest, coo3) + np.kron(step_plus, cxx3) + np.kron(step_minus, cyy3)
S_C3 = step_rest + step_plus + step_minus
#========== the Evolution operator ====================================
UH2 = S_Q2.dot(np.kron(np.eye(N), H2))
UG3 = S_Q3.dot(np.kron(np.eye(N), G3))
UF3 = S_Q3.dot(np.kron(np.eye(N), F3))
UC2 = S_C2/(2.)
UC3 = S_C3/(3.)
#========== Define the initial state Psi(zero) ===============
posn0 = np.zeros(N)
posn0[0] = 1
psi0Q2 = np.kron(posn0,((coin2[0])*X+(coin2[1])*Y)/np.sqrt(X+Y))
psi0C2 = posn0
psi0Q3 = np.kron(posn0,((coin3[0]*O)+(coin3[1]*X)+(coin3[2]*Y))/np.sqrt(X+Y+O))
psi0C3 = posn0
#============================
yy = (T+1)
Arriva_probH2 = np.zeros(yy)
Arriva_probG3 = np.zeros(yy)
Arriva_probF3 = np.zeros(yy)
Arriva_probC2 = np.zeros(yy)
Arriva_probC3 = np.zeros(yy)
Norm_psiNH2 = np.zeros(yy)
Norm_psiNC2 = np.zeros(yy)
```

```
Norm_psiNG3 = np.zeros(yy)
Norm_psiNF3 = np.zeros(yy)
Norm_psiNC3 = np.zeros(yy)
aH2 =0
aC2 =0
aG3 =0
aF3 =0
aC3 =0
bH2 = np.zeros(yy)
bC2 = np.zeros(yy)
bG3 = np.zeros(yy)
bF3 = np.zeros(yy)
bC3 = np.zeros(yy)
#============================================================
t=0
while t<=T:
        if t==0:
                psiNH2 = psi0Q2
                psiNC2 = psi0C2
                psiNF3 = psi0Q3
                psiNG3 = psi0Q3
                psiNC3 = psi0C3
        else :
                psiNH2 = np.dot(UH2, psiNH2)
                psiNC2 = np.dot(UC2, psiNC2
                psiNF3 = np.dot(UF3, psiNF3)
                psiNG3 = np.dot(UG3, psiNG3)
                psiNC3 = np.dot(UC3, psiNC3)
        #===Hadamard: 2D===================
        psiNH2[(Lev-1)*2] = 0
        psiNH2[((Lev-1)*2)+1] = 0
        #--------------------------------------
        Norm_psiNH2[t] = psiNH2.dot(psiNH2.conjugate()).real
        Arriva_probH2[t] = 1 - Norm_psiNH2[t]
        #===Grover: 3D====================
        psiNG3[(Lev-1)*3] = 0
        psiNG3[((Lev-1)*3)+1] = 0
        psiNG3[((Lev-1)*3)+2] = 0
        #----------------------
        Norm_psiNG3[t] = psiNG3.dot(psiNG3.conjugate()).real
        Arriva_probG3[t] = 1 - Norm_psiNG3[t]
        #===Fourier: 3D====================
        psiNF3[(Lev-1)*3] = 0
        psiNF3[((Lev-1)*3)+1] = 0
        psiNF3[((Lev-1)*3)+2] = 0
        #----------------------
        Norm_psiNF3[t] = psiNF3.dot(psiNF3.conjugate()).real
        Arriva_probF3[t] = 1 - Norm_psiNF3[t]
        #===classic: 2D ====================
        aC2 = aC2 + psiNC2[(Lev-1)]
        Arriva_probC2[t] = aC2
        psiNC2[(Lev-1)] = 0
        #===classic: 3D ====================
        aC3 = aC3 + psiNC3[(Lev-1)]
        Arriva_probC3[t] = aC3
        psiNC3[(Lev-1)] = 0
        t+=1
t-=1
#===== the plotted part ======================================
minorLocator = AutoMinorLocator()
fig, ax = plt.subplots()
plt.plot(Arriva_probH2, 'r-', label='H 2D', markersize=20, linewidth=7)
plt.plot(Arriva_probG3, 'b--', label='G 3D', markersize=20, linewidth=7, c='lime')
plt.plot(Arriva_probF3, 'b:', label='F 3D', markersize=20, linewidth=7)
plt.plot(Arriva_probC2, 'k-', label='C 2D', markersize=20, linewidth=7)
plt.plot(Arriva_probC3, 'k--', label='C 3D', markersize=20, linewidth=7)
plt.xticks(np.arange (0, yy, 20))
plt.xlim(0, yy)
plt.ylim(0, 1)
ax.xaxis.set_minor_locator(minorLocator)
ax.yaxis.set_minor_locator(minorLocator)
ax.grid(which='both')
ax.grid(which='minor', lw=0.5, ls=':')
ax.grid(which='major', lw=1, ls='--')
ax.tick_params(axis='x', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='x', which='minor', labelsize=70, width=2, length=10, colors='k')
ax.tick_params(axis='y', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='y', which='minor', labelsize=70, width=2, length=10, colors='k')
ax.set_xlabel('Steps number', fontsize=65)
ax.set_ylabel('Arrival Probability', color='k', fontsize=65)
for tl in ax.get_yticklabels():
tl.set_color('k')
ax.xaxis.set_minor_locator(MultipleLocator(50))
```

```
ax.yaxis.set_minor_locator(MultipleLocator(0.10))
plt.title('Arrival probability.circle '+str(Lev)+' levels, initial state(node 1, and equal mixture all coin state) .after
'+str(T)+' steps', fontsize=16)
plt.legend(fontsize=30)
figManager = plt.get_current_fig_manager()
figManager.window.showMaximized()
plt.tight_layout()
plt.show()
print ('الحمد لله')
#------the End of the code ---------------------------------------------------------------------------------------
```

البرنامج 3: تطور الاحتمال الواصل لجزيء C_{60} انطلاقا من عقد منفردة

```
print ('بسم الله الرحمان الرحيم')
#Author: Hamza Bougroura
#======= Import the requirement module ======================
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
#===define the critical number ==============================
T = 700   # number of random steps
N = 60    # number of positions in space (nodes)
Lev = 10  # Level of graph
#=== define the operators ===================================
G3 = (np.array([[-1, 2, 2], [2, -1, 2], [2, 2, -1]]))/(3.)
G4 = (np.array([[-2, 2, 2, 2], [2, -2, 2, 2], [2, 2, -2, 2], [2, 2, 2, -2]]))/(4.)
F3 = (np.fft.fft(np.eye(3)))/np.sqrt(3.)
F4 = (np.fft.fft(np.eye(4)))/np.sqrt(4.)
H4 = (np.array([[1, 1, 1, 1], [1, -1, 1, -1], [1, 1, -1, -1], [1, -1, -1, 1]]))/(2.)
#========== Define the coni states ==========================================
coin3 = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
coin4 = np.array([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])
#=====generate nodes states ==================================================
node = np.zeros((N+1, N))
k=0
while k<N:
        space = np.zeros(N)
        space[k] = 1
        node[k+1] = space
        k+=1
#====generate possible steps ================================
os1 = np.outer(node[1], node[1]) +np.outer(node[2], node[2]) +np.outer(node[3], node[3]) +np.outer(node[4], node[4])
+np.outer(node[5], node[5]) +np.outer(node[6], node[6])
os2 = os1 + np.outer(node[7], node[7])  +np.outer(node[8], node[8])  +np.outer(node[9], node[9])  +np.outer(node[10],
node[10])+np.outer(node[11], node[11])+np.outer(node[12], node[12])
os3 = os2 + np.outer(node[13], node[13])+np.outer(node[14], node[14])+np.outer(node[15], node[15])+np.outer(node[16],
node[16])+np.outer(node[17], node[17])+np.outer(node[18], node[18])
os4 = os3 + np.outer(node[19], node[19])+np.outer(node[20], node[20])+np.outer(node[21], node[21])+np.outer(node[22],
node[22])+np.outer(node[23], node[23])+np.outer(node[24], node[24])
os5 = os4 + np.outer(node[25], node[25])+np.outer(node[26], node[26])+np.outer(node[27], node[27])+np.outer(node[28],
node[28])+np.outer(node[29], node[29])+np.outer(node[30], node[30])
os6 = os5 + np.outer(node[31], node[31])+np.outer(node[32], node[32])+np.outer(node[33], node[33])+np.outer(node[34],
node[34])+np.outer(node[35], node[35])+np.outer(node[36], node[36])
os7 = os6 + np.outer(node[37], node[37])+np.outer(node[38], node[38])+np.outer(node[39], node[39])+np.outer(node[40],
node[40])+np.outer(node[41], node[41])+np.outer(node[42], node[42])
os8 = os7 + np.outer(node[43], node[43])+np.outer(node[44], node[44])+np.outer(node[45], node[45])+np.outer(node[46],
node[46])+np.outer(node[47], node[47])+np.outer(node[48], node[48])
os9 = os8 + np.outer(node[49], node[49])+np.outer(node[50], node[50])+np.outer(node[51], node[51])+np.outer(node[52],
node[52])+np.outer(node[53], node[53])+np.outer(node[54], node[54])
os10= os9 + np.outer(node[55], node[55])+np.outer(node[56], node[56])+np.outer(node[57], node[57])+np.outer(node[58],
node[58])+np.outer(node[59], node[59])+np.outer(node[60], node[60])
# os10: represent the possible jumping in space corresponding to |O><O|

xs1 = np.outer(node[3], node[1]) + np.outer(node[5], node[2]) + np.outer(node[1], node[3]) + np.outer(node[9], node[4])
+np.outer(node[2], node[5]) + np.outer(node[12], node[6])
xs2 = xs1 + np.outer(node[8], node[7])  +np.outer(node[7], node[8])  +np.outer(node[4], node[9])  +np.outer(node[17],
node[10])+np.outer(node[19], node[11])+np.outer(node[6], node[12])
xs3 = xs2 + np.outer(node[22], node[13])+np.outer(node[23], node[14])+np.outer(node[24], node[15])+np.outer(node[26],
node[16])+np.outer(node[10], node[17])+np.outer(node[28], node[18])
xs4 = xs3 + np.outer(node[11], node[19])+np.outer(node[30], node[20])+np.outer(node[31], node[21])+np.outer(node[13],
node[22])+np.outer(node[14], node[23])+np.outer(node[15], node[24])
xs5 = xs4 + np.outer(node[35], node[25])+np.outer(node[16], node[26])+np.outer(node[37], node[27])+np.outer(node[18],
node[28])+np.outer(node[38], node[29])+np.outer(node[20], node[30])
xs6 = xs5 + np.outer(node[21], node[31])+np.outer(node[42], node[32])+np.outer(node[43], node[33])+np.outer(node[44],
node[34])+np.outer(node[25], node[35])+np.outer(node[46], node[36])
xs7 = xs6 + np.outer(node[27], node[37])+np.outer(node[29], node[38])+np.outer(node[40], node[39])+np.outer(node[39],
node[40])+np.outer(node[50], node[41])+np.outer(node[32], node[42])
xs8 = xs7 + np.outer(node[33], node[43])+np.outer(node[34], node[44])+np.outer(node[36], node[46])+np.outer(node[48],
node[47])+np.outer(node[47], node[48])
xs9 = xs8 + np.outer(node[57], node[49])+np.outer(node[41], node[50])+np.outer(node[52], node[51])+np.outer(node[51],
node[52])+np.outer(node[58], node[53])+np.outer(node[55], node[54])
```

```
xs10= xs9 + np.outer(node[54], node[55])+np.outer(node[59], node[56])+np.outer(node[49], node[57])+np.outer(node[53],
node[58])+np.outer(node[56], node[59])
# xs10: represent the possible jumping in space corresponding to |X><X|

ys1 = np.outer(node[4], node[1])  +np.outer(node[10], node[2]) +np.outer(node[7], node[3])  +np.outer(node[1], node[4])
+np.outer(node[12], node[5]) +np.outer(node[13], node[6])
ys2 = ys1 + np.outer(node[3], node[7])  + np.outer(node[15], node[8]) + np.outer(node[17], node[9]) + np.outer(node[2],
node[10]) + np.outer(node[18], node[11])+ np.outer(node[5], node[12])
ys3 = ys2 + np.outer(node[6], node[13]) +np.outer(node[22], node[14])+np.outer(node[8], node[15]) +np.outer(node[25],
node[16])+np.outer(node[9], node[17]) +np.outer(node[11], node[18])
ys4 = ys3 + np.outer(node[30], node[19])+np.outer(node[21], node[20])+np.outer(node[20], node[21])+np.outer(node[14],
node[22])+np.outer(node[24], node[23])+np.outer(node[23], node[24])
ys5 = ys4 + np.outer(node[16], node[25])+np.outer(node[27], node[26])+np.outer(node[26], node[27])+np.outer(node[37],
node[28])+np.outer(node[39], node[29])+np.outer(node[19], node[30])
ys6 = ys5 + np.outer(node[42], node[31])+np.outer(node[33], node[32])+np.outer(node[32], node[33])+np.outer(node[35],
node[34])+np.outer(node[34], node[35])+np.outer(node[45], node[36])
ys7 = ys6 + np.outer(node[28], node[37])+np.outer(node[48], node[38])+np.outer(node[29], node[39])+np.outer(node[41],
node[40])+np.outer(node[40], node[41])+np.outer(node[31], node[42])
ys8 = ys7 + np.outer(node[44], node[43])+np.outer(node[43], node[44])+np.outer(node[36], node[45])+np.outer(node[47],
node[46])+np.outer(node[46], node[47])+np.outer(node[38], node[48])
ys9 = ys8 + np.outer(node[56], node[49])+np.outer(node[51], node[50])+np.outer(node[50], node[51])+np.outer(node[58],
node[52])+np.outer(node[54], node[53])+np.outer(node[53], node[54])
ys10= ys9 + np.outer(node[59], node[55])+np.outer(node[49], node[56])+np.outer(node[60], node[57])+np.outer(node[52],
node[58])+np.outer(node[55], node[59])+np.outer(node[57], node[60])
# ys10: represent the possible jumping in space corresponding to |Y><Y|

zs1 = np.outer(node[2], node[1]) +np.outer(node[1], node[2])+np.outer(node[6], node[3])  +np.outer(node[8], node[4])
+np.outer(node[11], node[5]) +np.outer(node[3], node[6])
zs2 = zs1 + np.outer(node[14], node[7]) +np.outer(node[4], node[8])  +np.outer(node[16], node[9]) +np.outer(node[18],
node[10])+np.outer(node[5], node[11]) +np.outer(node[20], node[12])
zs3 = zs2 + np.outer(node[21], node[13])+np.outer(node[7], node[14]) +np.outer(node[25], node[15])+ np.outer(node[9],
node[16]) +np.outer(node[27], node[17])+np.outer(node[10], node[18])
zs4 = zs3 + np.outer(node[29], node[19])+np.outer(node[12], node[20])+np.outer(node[13], node[21])+np.outer(node[32],
node[22])+np.outer(node[33], node[23])+np.outer(node[34], node[24])
zs5 = zs4 + np.outer(node[15], node[25])+np.outer(node[36], node[26])+np.outer(node[17], node[27])+np.outer(node[38],
node[28])+np.outer(node[19], node[29])+np.outer(node[40], node[30])
zs6 = zs5 + np.outer(node[41], node[31])+np.outer(node[22], node[32])+np.outer(node[23], node[33])+np.outer(node[24],
node[34])+np.outer(node[45], node[35])+np.outer(node[26], node[36])
zs7 = zs6 + np.outer(node[47], node[37])+np.outer(node[28], node[38])+np.outer(node[49], node[39])+np.outer(node[30],
node[40])+np.outer(node[31], node[41])+np.outer(node[51], node[42])
zs8 = zs7 + np.outer(node[52], node[43])+np.outer(node[53], node[44])+np.outer(node[35], node[45])+np.outer(node[55],
node[46])+np.outer(node[37], node[47])+np.outer(node[56], node[48])
zs9 = zs8 + np.outer(node[39], node[49])+np.outer(node[57], node[50])+np.outer(node[42], node[51])+np.outer(node[43],
node[52])+np.outer(node[44], node[53])
zs10= zs9 + np.outer(node[46], node[55])+np.outer(node[48], node[56])+np.outer(node[50], node[57])+np.outer(node[60],
node[58])+ np.outer(node[58], node[60])
# zs10: represent the possible jumping in space corresponding to |Z><Z|

# the except four jumping space are(corresponding to |Z><X| and |X><Z|)
node54_45= np.outer(node[54], node[45])
node45_54= np.outer(node[45], node[54])
node60_59= np.outer(node[60], node[59])
node59_60= np.outer(node[59], node[60])
# the coin state terms that we need without change (in 4D)
coo4 = np.outer(coin4[0], coin4[0])
cxx4 = np.outer(coin4[1], coin4[1])
cyy4 = np.outer(coin4[2], coin4[2])
czz4 = np.outer(coin4[3], coin4[3])
# the coin state terms that we need without change (in 3D)
cxx3 = np.outer(coin3[0], coin3[0])
cyy3 = np.outer(coin3[1], coin3[1])
czz3 = np.outer(coin3[2], coin3[2])
# the coin state terms that we need with change (in 4D)
czx4 = np.outer(coin4[3], coin4[1])
cxz4 = np.outer(coin4[1], coin4[3])
# the coin state terms that we need with change (in 3D)
czx3 = np.outer(coin3[2], coin3[0])
cxz3 = np.outer(coin3[0], coin3[2])
# in the following terms there is not change in the coin state when the jumping in space is happenig (4D)
S_hat_O4 = np.kron(os10, coo4)
S_hat_X4 = np.kron(xs10, cxx4)
S_hat_Y4 = np.kron(ys10, cyy4)
S_hat_Z4 = np.kron(zs10, czz4)
# in the following terms there is not change in the coin state when the jumping in space is happenig (3D)
S_hat_X3 = np.kron(xs10, cxx3)
S_hat_Y3 = np.kron(ys10, cyy3)
S_hat_Z3 = np.kron(zs10, czz3)
# However there is only four terms that we should change the coin state when the jumping in space is happening (4D)
zx4_54_45 = np.kron(node54_45, czx4)
xz4_45_54 = np.kron(node45_54, cxz4)
xz4_60_59 = np.kron(node60_59, cxz4)
zx4_59_60 = np.kron(node59_60, czx4)
# However there is only four terms that we should change the coin state when the jumping in space is happening (3D)
```

```
zx3_54_45 = np.kron(node54_45, czx3)
xz3_45_54 = np.kron(node45_54, cxz3)
xz3_60_59 = np.kron(node60_59, cxz3)
zx3_59_60 = np.kron(node59_60, czx3)
# Finally the Shift operator is:
S_hatQ4 = S_hat_O4 + S_hat_X4 + S_hat_Y4 + S_hat_Z4 + zx4_54_45 + xz4_45_54 + xz4_60_59 + zx4_59_60
S_hatQ3 = S_hat_X3 + S_hat_Y3 + S_hat_Z3 + zx3_54_45 + xz3_45_54 + xz3_60_59 + zx3_59_60
S_hatC4 = os10 + xs10 + ys10 + zs10 + node54_45 + node45_54 + node60_59 + node59_60
S_hatC3 = xs10 + ys10 + zs10 + node54_45 + node45_54 + node60_59 + node59_60
#======= Evolution operator ======================
UH4 = S_hatQ4.dot(np.kron(np.eye(N), H4))
UG4 = S_hatQ4.dot(np.kron(np.eye(N), G4))
UF4 = S_hatQ4.dot(np.kron(np.eye(N), F4))
UG3 = S_hatQ3.dot(np.kron(np.eye(N), G3))
UF3 = S_hatQ3.dot(np.kron(np.eye(N), F3))
UC4= S_hatC4/4
UC3= S_hatC3/3
#======define the initial state Psi(zero)============================
psi0Q4 = np.kron(node[1],(coin4[0]+coin4[1]+coin4[2]+coin4[3])/np.sqrt(4.))
psi0Q3 = np.kron(node[1],(coin3[0]+coin3[1]+coin3[2])/np.sqrt(3.))
psi0C4 = node[1]
psi0C3 = node[1]
#-------------------------------------------------------------------------
yy= T+1
Arri_Prob_H4 = np.zeros(yy)
Arri_Prob_G3 = np.zeros(yy)
Arri_Prob_G4 = np.zeros(yy)
Arri_Prob_F3 = np.zeros(yy)
Arri_Prob_F4 = np.zeros(yy)
Arri_Prob_C3 = np.zeros(yy)
Arri_Prob_C4 = np.zeros(yy)
Norm_psi_t_H4 = np.zeros(yy)
Norm_psi_t_G3 = np.zeros(yy)
Norm_psi_t_G4 = np.zeros(yy)
Norm_psi_t_F3 = np.zeros(yy)
Norm_psi_t_F4 = np.zeros(yy)
Norm_psi_t_C3 = np.zeros(yy)
Norm_psi_t_C4 = np.zeros(yy)
aH4 =0
aG4 =0
aG3 =0
aF4 =0
aF3 =0
aC4 =0
aC3 =0
bH4 = np.zeros(yy)
bG4 = np.zeros(yy)
bG3 = np.zeros(yy)
bF4 = np.zeros(yy)
bF3 = np.zeros(yy)
bC4 = np.zeros(yy)
bC3 = np.zeros(yy)
#-------------------------------------------------------------------------
t=0
while t<=T:
        if t==0:
                psi_t_G3 = psi0Q3
                psi_t_G4 = psi0Q4
                psi_t_F3 = psi0Q3
                psi_t_F4 = psi0Q4
                psi_t_H4 = psi0Q4
                psi_t_C3 = psi0C3
                psi_t_C4 = psi0C4
        else :
                psi_t_G4 = np.dot(UG4, psi_t_G4)
                psi_t_G3 = np.dot(UG3, psi_t_G3)
                psi_t_F3 = np.dot(UF3, psi_t_F3)
                psi_t_F4 = np.dot(UF4, psi_t_F4)
                psi_t_H4 = np.dot(UH4, psi_t_H4)
                psi_t_C4 = np.dot(UC4, psi_t_C4)
                psi_t_C3 = np.dot(UC3, psi_t_C3)
        #====Grover 3D=================================
        e3 = ((59)*3)
        while e3<(60*3):
        psi_t_G3[e3] = 0
        e3+=1
        #-----------------------------------------
        Norm_psi_t_G3[t] = psi_t_G3.dot(psi_t_G3.conjugate()).real
        Arri_Prob_G3[t] = 1 - Norm_psi_t_G3[t]
        #====Grover 4D=================================
        e4 = ((59)*4)
        while e4<(60*4):
        psi_t_G4[e4] = 0
```

```
        e4+=1
        #-------------------------------------------
        Norm_psi_t_G4[t] = psi_t_G4.dot(psi_t_G4.conjugate()).real
        Arri_Prob_G4[t] = 1 - Norm_psi_t_G4[t]
        #====Fourier 3D=================================
        e3 = ((59)*3)
        while e3<(60*3):
        psi_t_F3[e3] = 0
        e3+=1
        #-------------------------------------------
        Norm_psi_t_F3[t] = psi_t_F3.dot(psi_t_F3.conjugate()).real
        Arri_Prob_F3[t] = 1 - Norm_psi_t_F3[t]
        #====Fourier 4D=================================
        e4 = ((59)*4)
        while e4<(60*4):
        psi_t_F4[e4] = 0
        e4+=1
        #-------------------------------------------
        Norm_psi_t_F4[t] = psi_t_F4.dot(psi_t_F4.conjugate()).real
        Arri_Prob_F4[t] = 1 - Norm_psi_t_F4[t]
        #========== Hadamard 4D=========================
        e4 = ((59)*4)
        while e4<(60*4):
        psi_t_H4[e4] = 0
        e4+=1
        #-------------------------------------------
        Norm_psi_t_H4[t] = psi_t_H4.dot(psi_t_H4.conjugate()).real
        Arri_Prob_H4[t] = 1 - Norm_psi_t_H4[t]
        #=========classical case 4D=====================
        psi_t_C4[59]=0
        #---------------------------------
        Norm_psi_t_C4[t] = np.sum(psi_t_C4)
        Arri_Prob_C4[t] = 1 - Norm_psi_t_C4[t]
        #=========classical case 3D=====================
        psi_t_C3[59]=0
        #-------------------------------------------
        Norm_psi_t_C3[t] = np.sum(psi_t_C3)
        Arri_Prob_C3[t] = 1 - Norm_psi_t_C3[t]
        t+=1
t-=1
#======= the plotting part of the arrival probability vector===================
minorLocator = AutoMinorLocator()
fig, ax = plt.subplots()
plt.plot(Arri_Prob_G3, 'g-', label='G 3D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_G4, 'g--', label='G 4D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_F3, 'c-', label='F 3D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_F4, 'c--', label='F 4D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_H4, 'y--', label='H 4D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_C3, 'k-', label='C 3D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_C4, 'k--', label='C 4D', markersize=12, linewidth=7)
ax.xaxis.set_minor_locator(minorLocator)
ax.tick_params(axis='x', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='x', which='minor', labelsize=70, width=2, length=20, colors='k')
ax.tick_params(axis='y', which='major', labelsize=70, width=2, length=20, colors='k')
ax.set_xlabel('Steps number', fontsize=65)
ax.set_ylabel('Arrival probability', color='k', fontsize=65)
for tl in ax.get_yticklabels():
tl.set_color('k')
plt.xlim(0, yy)
plt.ylim(0, 1)
plt.title('Arrival probability.C60.initial state (Single atom, and equel mixture of all coin stetes, after ' + str(T) + ' steps',
fontsize=25)
plt.legend(fontsize=20, loc='best', borderaxespad=0.)
figManager = plt.get_current_fig_manager()
figManager.window.showMaximized()
plt.tight_layout()
plt.show()
print ('الحمد لله')
#------the End of the code ---------------------------------------------------------------------------------------------
```

البرنامج 4: تطور الاحتمال الواصل لجزيء C_{60} انطلاقا من مضلع خماسي

```
print('بسم الله الرحمان الرحيم')
#Author: Hamza Bougroura
#=== Import the requirement module ===============
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
#===Define the critical number ====================
T = 200 # number of random steps
N = 60 # number of positions in space (nodes)
```

```
Lev = 8 # Level of graph
#===Define Grover coin operator 3D =================
G3 = (np.array([[-1, 2, 2], [2, -1, 2], [2, 2, -1]]))/(3.)
#=====generate coin state ===================================
coin3 = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
#=====generate space state that represent nodes ================
node = np.zeros((N+1, N))
k=0
while k<N:
        space = np.zeros(N)
        space[k] = 1
        node[k+1] = space
        k+=1
#====generate possible steps ==================================
os1 =       np.outer(node[1], node[1])  +np.outer(node[2], node[2])  +np.outer(node[3], node[3])  +np.outer(node[4], node[4])
+np.outer(node[5], node[5])  +np.outer(node[6], node[6])
os2 = os1 + np.outer(node[7], node[7])   +np.outer(node[8], node[8])   +np.outer(node[9], node[9])   +np.outer(node[10],
node[10])+np.outer(node[11], node[11])+np.outer(node[12], node[12])
os3 = os2 + np.outer(node[13], node[13])+np.outer(node[14], node[14])+np.outer(node[15], node[15])+np.outer(node[16],
node[16])+np.outer(node[17], node[17])+np.outer(node[18], node[18])
os4 = os3 + np.outer(node[19], node[19])+np.outer(node[20], node[20])+np.outer(node[21], node[21])+np.outer(node[22],
node[22])+np.outer(node[23], node[23])+np.outer(node[24], node[24])
os5 = os4 + np.outer(node[25], node[25])+np.outer(node[26], node[26])+np.outer(node[27], node[27])+np.outer(node[28],
node[28])+np.outer(node[29], node[29])+np.outer(node[30], node[30])
os6 = os5 + np.outer(node[31], node[31])+np.outer(node[32], node[32])+np.outer(node[33], node[33])+np.outer(node[34],
node[34])+np.outer(node[35], node[35])+np.outer(node[36], node[36])
os7 = os6 + np.outer(node[37], node[37])+np.outer(node[38], node[38])+np.outer(node[39], node[39])+np.outer(node[40],
node[40])+np.outer(node[41], node[41])+np.outer(node[42], node[42])
os8 = os7 + np.outer(node[43], node[43])+np.outer(node[44], node[44])+np.outer(node[45], node[45])+np.outer(node[46],
node[46])+np.outer(node[47], node[47])+np.outer(node[48], node[48])
os9 = os8 + np.outer(node[49], node[49])+np.outer(node[50], node[50])+np.outer(node[51], node[51])+np.outer(node[52],
node[52])+np.outer(node[53], node[53])+np.outer(node[54], node[54])
os10= os9 + np.outer(node[55], node[55])+np.outer(node[56], node[56])+np.outer(node[57], node[57])+np.outer(node[58],
node[58])+np.outer(node[59], node[59])+np.outer(node[60], node[60])
# os10: represent the possible jumping in space corresponding to |O><O|

xs1 =        np.outer(node[2], node[1])  +np.outer(node[1], node[2])  +np.outer(node[4], node[3]) +np.outer(node[3], node[4])
+np.outer(node[10], node[5]) +np.outer(node[11], node[6])
xs2 = xs1 + np.outer(node[12], node[7])    +np.outer(node[15], node[8])+np.outer(node[16], node[9])+np.outer(node[5],
node[10])+np.outer(node[6], node[11]) +np.outer(node[7], node[12])
xs3 = xs2 + np.outer(node[14], node[13])+np.outer(node[13], node[14])+np.outer(node[8], node[15])+np.outer(node[9],
node[16])+np.outer(node[27], node[17])+np.outer(node[28], node[18])
xs4 = xs3 + np.outer(node[20], node[19])+np.outer(node[19], node[20])+np.outer(node[35], node[21])+np.outer(node[34],
node[22])+np.outer(node[33], node[23])+np.outer(node[32], node[24])
xs5 = xs4 + np.outer(node[31], node[25])+np.outer(node[40], node[26])+np.outer(node[17], node[27])+np.outer(node[18],
node[28])+np.outer(node[37], node[29])+np.outer(node[36], node[30])
xs6 = xs5 + np.outer(node[25], node[31])+np.outer(node[24], node[32])+np.outer(node[23], node[33])+np.outer(node[22],
node[34])+np.outer(node[21], node[35])+np.outer(node[30], node[36])
xs7 = xs6 + np.outer(node[29], node[37])+np.outer(node[48], node[38])+np.outer(node[26], node[40])+np.outer(node[42],
node[41])+np.outer(node[41], node[42])
xs8 = xs7 + np.outer(node[51], node[43])+np.outer(node[52], node[44])+np.outer(node[46], node[45])+np.outer(node[45],
node[46])+np.outer(node[53], node[47])+np.outer(node[38], node[48])
xs9 = xs8 + np.outer(node[54], node[49])+np.outer(node[55], node[50])+np.outer(node[43], node[51])+np.outer(node[44],
node[52])+np.outer(node[47], node[53])+np.outer(node[49], node[54])
xs10= xs9 + np.outer(node[50], node[55])+np.outer(node[57], node[56])+np.outer(node[56], node[57])+np.outer(node[59],
node[58])+np.outer(node[58], node[59])
# xs10: represent the possible jumping in space corresponding to |X><X|

ys1 =       np.outer(node[5], node[1])  +np.outer(node[3], node[2])  +np.outer(node[2], node[3])  +np.outer(node[9], node[4])
+np.outer(node[1], node[5])  +np.outer(node[20], node[6])
ys2 = ys1 + np.outer(node[13], node[7]) +np.outer(node[14], node[8]) +np.outer(node[4], node[9])   +np.outer(node[19],
node[10])+np.outer(node[12], node[11])+np.outer(node[11], node[12])
ys3 = ys2 + np.outer(node[7], node[13])  +np.outer(node[8], node[14])  +np.outer(node[16], node[15])+np.outer(node[15],
node[16])+np.outer(node[18], node[17])+np.outer(node[17], node[18])
ys4 = ys3 + np.outer(node[10], node[19])+np.outer(node[6], node[20])  +np.outer(node[30], node[21])+np.outer(node[23],
node[22])+np.outer(node[22], node[23])+np.outer(node[25], node[24])
ys5 = ys4 + np.outer(node[24], node[25])+np.outer(node[27], node[26])+np.outer(node[26], node[27])+np.outer(node[29],
node[28])+np.outer(node[28], node[29])+np.outer(node[21], node[30])
ys6 = ys5 + np.outer(node[40], node[31])+np.outer(node[33], node[32])+np.outer(node[32], node[33])+np.outer(node[35],
node[34])+np.outer(node[34], node[35])+np.outer(node[37], node[36])
ys7 = ys6 + np.outer(node[36], node[37])+np.outer(node[39], node[38])+np.outer(node[38], node[39])+np.outer(node[31],
node[40])+np.outer(node[55], node[41])+np.outer(node[51], node[42])
ys8 = ys7 + np.outer(node[44], node[43])+np.outer(node[43], node[44])+np.outer(node[52], node[45])+np.outer(node[53],
node[46])+np.outer(node[48], node[47])+np.outer(node[47], node[48])
ys9 = ys8 + np.outer(node[50], node[49])+np.outer(node[49], node[50])+np.outer(node[42], node[51])+np.outer(node[45],
node[52])+np.outer(node[46], node[53])+np.outer(node[59], node[54])
ys10= ys9 + np.outer(node[41], node[55])+np.outer(node[60], node[56])+np.outer(node[58], node[57])+np.outer(node[57],
node[58])+np.outer(node[54], node[59])+np.outer(node[56], node[60])
# ys10: represent the possible jumping in space corresponding to |Y><Y|

zs1 =       np.outer(node[6], node[1])  +np.outer(node[7], node[2])  +np.outer(node[8], node[3])  +np.outer(node[5], node[4])
+np.outer(node[4], node[5])  +np.outer(node[1], node[6])
```

```
zs2 = zs1 + np.outer(node[2], node[7])   +np.outer(node[3], node[8])   +np.outer(node[17], node[9]) +np.outer(node[18], node[10])+np.outer(node[21], node[11])+np.outer(node[22], node[12])
zs3 = zs2 + np.outer(node[23], node[13])+np.outer(node[24], node[14])+np.outer(node[25], node[15])+np.outer(node[26], node[16])+np.outer(node[9], node[17]) +np.outer(node[10], node[18])
zs4 = zs3 + np.outer(node[29], node[19])+np.outer(node[30], node[20])+np.outer(node[11], node[21])+np.outer(node[12], node[22])+np.outer(node[13], node[23])+np.outer(node[14], node[24])
zs5 = zs4 + np.outer(node[15], node[25])+np.outer(node[16], node[26])+np.outer(node[39], node[27])+np.outer(node[38], node[28])+np.outer(node[19], node[29])+np.outer(node[20], node[30])
zs6 = zs5 + np.outer(node[41], node[31])+np.outer(node[42], node[32])+np.outer(node[43], node[33])+np.outer(node[44], node[34])+np.outer(node[45], node[35])+np.outer(node[46], node[36])
zs7 = zs6 + np.outer(node[47], node[37])+np.outer(node[28], node[38])+np.outer(node[27], node[39])+np.outer(node[50], node[40])+np.outer(node[31], node[41])+np.outer(node[32], node[42])
zs8 = zs7 + np.outer(node[33], node[43])+np.outer(node[34], node[44])+np.outer(node[35], node[45])+np.outer(node[36], node[46])+np.outer(node[37], node[47])+np.outer(node[54], node[48])
zs9 = zs8 + np.outer(node[40], node[50])+np.outer(node[56], node[51])+np.outer(node[57], node[52])+np.outer(node[58], node[53])+np.outer(node[48], node[54])
zs10= zs9 + np.outer(node[60], node[55])+np.outer(node[51], node[56])+np.outer(node[52], node[57])+np.outer(node[53], node[58])+np.outer(node[55], node[60])
# zs10: represent the possible jumping in space corresponding to |Z><Z|

# the except four jumping space are(corresponding to |Z><X| and |X><Z|)
node49_39= np.outer(node[49], node[39])
node39_49= np.outer(node[39], node[49])
node60_59= np.outer(node[60], node[59])
node59_60= np.outer(node[59], node[60])
# the coin state terms that we need without change (in 3D)
cxx3 = np.outer(coin3[0], coin3[0])
cyy3 = np.outer(coin3[1], coin3[1])
czz3 = np.outer(coin3[2], coin3[2])
# the coin state terms that we need with change (in 3D)
czx3 = np.outer(coin3[2], coin3[0])
cxz3 = np.outer(coin3[0], coin3[2])
# in the following terms there is not change in the coin state when the jumping in space is happenig (3D)
S_hat_X3 = np.kron(xs10, cxx3)
S_hat_Y3 = np.kron(ys10, cyy3)
S_hat_Z3 = np.kron(zs10, czz3)
# However there is only four terms that we should change the coin state when the jumping in space is happening (3D)
zx3_49_39 = np.kron(node49_39, czx3)
xz3_39_49 = np.kron(node39_49, cxz3)
xz3_60_59 = np.kron(node60_59, cxz3)
zx3_59_60 = np.kron(node59_60, czx3)
# Finally the Shift operator is:
S_hatQ3 = S_hat_X3 + S_hat_Y3 + S_hat_Z3 + zx3_49_39 + xz3_39_49 + xz3_60_59 + zx3_59_60
S_hatC3 = xs10 + ys10 + zs10 + node49_39 + node39_49 + node60_59 + node59_60
#===== Evolution operator ======
UG3 = S_hatQ3.dot(np.kron(np.eye(N), G3))
UC3 = S_hatC3/3
#===Define the initial system state psi(zero)=====
psi_0_Q3 = np.kron((node[1]+node[2]+node[3]+node[4]+node[5])/np.sqrt(5.),(coin3[0]+coin3[1]+coin3[2])/np.sqrt(3.))
psi_0_C3 =(node[1]+node[2]+node[3]+node[4]+node[5])/(5.)
#=====================================================
yy= T+1
Arri_Prob_G3 = np.zeros(yy)
Arri_Prob_C3 = np.zeros(yy)
aG3 =0
aC3 =0
bG3 = np.zeros(yy)
bC3 = np.zeros(yy)
Norm_psi_t_G3 = np.zeros(yy)
Norm_psi_t_C3 = np.zeros(yy)
#======================================
t=0
while t<=T:
        if t==0:
                psi_t_G3 = psi_0_Q3
                psi_t_C3 = psi_0_C3
        else :
                psi_t_G3 = np.dot(UG3, psi_t_G3)
                psi_t_C3 = np.dot(UC3, psi_t_C3)
        #====Grover 3D================================
        e3 = ((55)*3)
        while e3<(60*3):
        psi_t_G3[e3] = 0
        e3+=1
        #---------------------------------------------
        Norm_psi_t_G3[t] = psi_t_G3.dot(psi_t_G3.conjugate()).real
        Arri_Prob_G3[t] = 1- Norm_psi_t_G3[t]
        #=========classical case 3D=====================
        psi_t_C3[55]=0
        psi_t_C3[56]=0
        psi_t_C3[57]=0
        psi_t_C3[58]=0
        psi_t_C3[59]=0
```

```
        #-------------------------------------------
        Norm_psi_t_C3[t] = np.sum(psi_t_C3)
        Arri_Prob_C3[t] = 1 - Norm_psi_t_C3[t]
        #-----------------------------------------------------------
        t+=1
t-=1
#====the plotting part =================
minorLocator = AutoMinorLocator()
fig, ax = plt.subplots()
plt.plot(Arri_Prob_G3, 'g-', label='G 3D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_C3, 'k-', label='C 3D', markersize=12, linewidth=7)
ax.xaxis.set_minor_locator(minorLocator)
ax.tick_params(axis='x', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='x', which='minor', labelsize=70, width=2, length=20, colors='k')
ax.tick_params(axis='y', which='major', labelsize=70, width=2, length=20, colors='k')
ax.set_xlabel('Steps number', fontsize=65)
ax.set_ylabel('Arrival probability', color='k', fontsize=65)
for tl in ax.get_yticklabels():
tl.set_color('k')
plt.xlim(0, yy)
plt.ylim(0, 1)
plt.title('Arrival probability,C06. 8 Levels, starting from 5 vertex and and equal mixture of all coin states.after ' + str(T)
+ ' steps', fontsize=16)
plt.legend(fontsize=20, loc='best', borderaxespad=0.)
figManager = plt.get_current_fig_manager()
figManager.window.showMaximized()
plt.tight_layout()
plt.show()
print ('الحمد لله')
#------the End of the code ---------------------------------------------------------------------------------------------
```

البرنامج 5: تطور الاحتمال الواصل لجزيء C_{60} انطلاقا من مضلع سداسي

```
print('بسم الله الرحمان الرحيم')
#Author: Hamza Bougroura
#=== Import the requirement module ===============
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
#===Define the critical number ====================
T = 200 # number of random steps
N = 60 # number of positions in space (nodes)
Lev = 8 # Level of graph
#===Define Grover coin operator 3D =================
G3 = (np.array([[-1, 2, 2], [2, -1, 2], [2, 2, -1]]))/(3.)
#=====generate coin state ===================================
coin3 = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
#=====generate space state that represent nodes ===============
node = np.zeros((N+1, N))
k=0
while k<N:
        space = np.zeros(N)
        space[k] = 1
        node[k+1] = space
        k+=1
#====generate possible steps ==================================
os1 =       np.outer(node[1], node[1])  +np.outer(node[2], node[2])  +np.outer(node[3], node[3])  +np.outer(node[4], node[4])
+np.outer(node[5], node[5])  +np.outer(node[6], node[6])
os2 = os1 + np.outer(node[7], node[7])   +np.outer(node[8], node[8])   +np.outer(node[9], node[9])   +np.outer(node[10],
node[10])+np.outer(node[11], node[11])+np.outer(node[12], node[12])
os3 = os2 + np.outer(node[13], node[13])+np.outer(node[14], node[14])+np.outer(node[15], node[15])+np.outer(node[16],
node[16])+np.outer(node[17], node[17])+np.outer(node[18], node[18])
os4 = os3 + np.outer(node[19], node[19])+np.outer(node[20], node[20])+np.outer(node[21], node[21])+np.outer(node[22],
node[22])+np.outer(node[23], node[23])+np.outer(node[24], node[24])
os5 = os4 + np.outer(node[25], node[25])+np.outer(node[26], node[26])+np.outer(node[27], node[27])+np.outer(node[28],
node[28])+np.outer(node[29], node[29])+np.outer(node[30], node[30])
os6 = os5 + np.outer(node[31], node[31])+np.outer(node[32], node[32])+np.outer(node[33], node[33])+np.outer(node[34],
node[34])+np.outer(node[35], node[35])+np.outer(node[36], node[36])
os7 = os6 + np.outer(node[37], node[37])+np.outer(node[38], node[38])+np.outer(node[39], node[39])+np.outer(node[40],
node[40])+np.outer(node[41], node[41])+np.outer(node[42], node[42])
os8 = os7 + np.outer(node[43], node[43])+np.outer(node[44], node[44])+np.outer(node[45], node[45])+np.outer(node[46],
node[46])+np.outer(node[47], node[47])+np.outer(node[48], node[48])
os9 = os8 + np.outer(node[49], node[49])+np.outer(node[50], node[50])+np.outer(node[51], node[51])+np.outer(node[52],
node[52])+np.outer(node[53], node[53])+np.outer(node[54], node[54])
os10= os9 + np.outer(node[55], node[55])+np.outer(node[56], node[56])+np.outer(node[57], node[57])+np.outer(node[58],
node[58])+np.outer(node[59], node[59])+np.outer(node[60], node[60])
# os10: represent the possible jumping in space corresponding to |O><O|

xs1 =       np.outer(node[6], node[1])  +np.outer(node[3], node[2])  +np.outer(node[2], node[3])  +np.outer(node[5], node[4])
+np.outer(node[4], node[5])  +np.outer(node[1], node[6])
xs2 = xs1 + np.outer(node[14], node[7]) +np.outer(node[15], node[8]) +np.outer(node[17], node[9]) +np.outer(node[18],
node[10])+np.outer(node[20], node[11])+np.outer(node[13], node[12])
```

```
xs3 = xs2 + np.outer(node[12], node[13])+np.outer(node[7], node[14]) +np.outer(node[8], node[15]) +np.outer(node[25], node[16])+np.outer(node[9], node[17]) +np.outer(node[10], node[18])
xs4 = xs3 + np.outer(node[28], node[19])+np.outer(node[11], node[20])+np.outer(node[30], node[21])+np.outer(node[35], node[22])+np.outer(node[33], node[23])+np.outer(node[32], node[24])
xs5 = xs4 + np.outer(node[16], node[25])+np.outer(node[31], node[26])+np.outer(node[39], node[27])+np.outer(node[19], node[28])+np.outer(node[37], node[29])+np.outer(node[21], node[30])
xs6 = xs5 + np.outer(node[26], node[31])+np.outer(node[24], node[32])+np.outer(node[23], node[33])+np.outer(node[43], node[34])+np.outer(node[22], node[35])+np.outer(node[45], node[36])
xs7 = xs6 + np.outer(node[29], node[37])+np.outer(node[47], node[38])+np.outer(node[27], node[39])+np.outer(node[41], node[40])+np.outer(node[40], node[41])+np.outer(node[50], node[42])
xs8 = xs7 + np.outer(node[34], node[43])+np.outer(node[52], node[44])+np.outer(node[36], node[45])+np.outer(node[53], node[46])+np.outer(node[38], node[47])+np.outer(node[54], node[48])
xs9 = xs8 + np.outer(node[55], node[49])+np.outer(node[42], node[50])+np.outer(node[44], node[52])+np.outer(node[46], node[53])+np.outer(node[48], node[54])
xs10= xs9 + np.outer(node[49], node[55])+np.outer(node[57], node[56])+np.outer(node[56], node[57])+np.outer(node[59], node[58])+np.outer(node[58], node[59])
# xs10: represent the possible jumping in space corresponding to |X><X|

ys1 = np.outer(node[7], node[1])+np.outer(node[8], node[2])+np.outer(node[4], node[3]) +np.outer(node[3], node[4]) +np.outer(node[11], node[5])+np.outer(node[12], node[6])
ys2 = ys1 + np.outer(node[1], node[7])+np.outer(node[2], node[8])+np.outer(node[16], node[9])+np.outer(node[19], node[10])+np.outer(node[5], node[11])+np.outer(node[6], node[12])
ys3 = ys2 + np.outer(node[22], node[13])+np.outer(node[15], node[14])+np.outer(node[14], node[15])+np.outer(node[9], node[16]) +np.outer(node[18], node[17])+np.outer(node[17], node[18])
ys4 = ys3 + np.outer(node[10], node[19])+np.outer(node[21], node[20])+np.outer(node[20], node[21])+np.outer(node[13], node[22])+np.outer(node[34], node[23])+np.outer(node[33], node[24])
ys5 = ys4 + np.outer(node[32], node[25])+np.outer(node[39], node[26])+np.outer(node[38], node[27])+np.outer(node[37], node[28])+np.outer(node[36], node[29])+np.outer(node[35], node[30])
ys6 = ys5 + np.outer(node[40], node[31])+np.outer(node[25], node[32])+np.outer(node[24], node[33])+np.outer(node[23], node[34])+np.outer(node[30], node[35])+np.outer(node[29], node[36])
ys7 = ys6 + np.outer(node[28], node[37])+np.outer(node[27], node[38])+np.outer(node[26], node[39])+np.outer(node[31], node[40])+np.outer(node[50], node[41])+np.outer(node[51], node[42])
ys8 = ys7 + np.outer(node[44], node[43])+np.outer(node[43], node[44])+np.outer(node[52], node[45])+np.outer(node[47], node[46])+np.outer(node[46], node[47])+np.outer(node[49], node[48])
ys9 = ys8 + np.outer(node[48], node[49])+np.outer(node[41], node[50])+np.outer(node[42], node[51])+np.outer(node[45], node[52])+np.outer(node[59], node[53])+np.outer(node[60], node[54])
ys10= ys9 + np.outer(node[56], node[55])+np.outer(node[55], node[56])+np.outer(node[58], node[57])+np.outer(node[57], node[58])+np.outer(node[53], node[59])+np.outer(node[54], node[60])
# ys10: represent the possible jumping in space corresponding to |Y><Y|

zs1 = np.outer(node[2], node[1]) +np.outer(node[1], node[2]) +np.outer(node[9], node[3]) +np.outer(node[10], node[4]) +np.outer(node[6], node[5]) +np.outer(node[5], node[6])
zs2 = zs1 + np.outer(node[13], node[7]) +np.outer(node[16], node[8]) +np.outer(node[3], node[9]) +np.outer(node[4], node[10]) +np.outer(node[19], node[11])+np.outer(node[21], node[12])
zs3 = zs2 + np.outer(node[7], node[13]) +np.outer(node[23], node[14])+np.outer(node[24], node[15])+np.outer(node[8], node[16]) +np.outer(node[26], node[17])+np.outer(node[27], node[18])
zs4 = zs3 + np.outer(node[11], node[19])+np.outer(node[29], node[20])+np.outer(node[12], node[21])+np.outer(node[34], node[22])+np.outer(node[14], node[23])+np.outer(node[15], node[24])
zs5 = zs4 + np.outer(node[31], node[25])+np.outer(node[17], node[26])+np.outer(node[18], node[27])+np.outer(node[38], node[28])+np.outer(node[20], node[29])+np.outer(node[36], node[30])
zs6 = zs5 + np.outer(node[25], node[31])+np.outer(node[41], node[32])+np.outer(node[42], node[33])+np.outer(node[22], node[34])+np.outer(node[44], node[35])+np.outer(node[30], node[36])
zs7 = zs6 + np.outer(node[46], node[37])+np.outer(node[28], node[38])+np.outer(node[48], node[39])+np.outer(node[49], node[40])+np.outer(node[32], node[41])+np.outer(node[33], node[42])
zs8 = zs7 + np.outer(node[51], node[43])+np.outer(node[35], node[44])+np.outer(node[53], node[45])+np.outer(node[37], node[46])+np.outer(node[54], node[47])+np.outer(node[39], node[48])
zs9 = zs8 + np.outer(node[40], node[49])+np.outer(node[56], node[50])+np.outer(node[43], node[51])+np.outer(node[58], node[52])+np.outer(node[45], node[53])+np.outer(node[47], node[54])
zs10= zs9 + np.outer(node[50], node[56])+np.outer(node[52], node[58])+np.outer(node[60], node[59])+np.outer(node[59], node[60])
# zs10: represent the possible jumping in space corresponding to |Z><Z|

# the except four jumping space are(corresponding to |Z><X| and |X><Z|)
node57_51= np.outer(node[57], node[51])
node51_57= np.outer(node[51], node[57])
node60_55= np.outer(node[60], node[55])
node55_60= np.outer(node[55], node[60])
# the coin state terms that we need without change (in 3D)
cxx3 = np.outer(coin3[0], coin3[0])
cyy3 = np.outer(coin3[1], coin3[1])
czz3 = np.outer(coin3[2], coin3[2])
# the coin state terms that we need with change (in 3D)
czx3 = np.outer(coin3[2], coin3[0])
cxz3 = np.outer(coin3[0], coin3[2])
# in the following terms there is not change in the coin state when the jumping in space is happenig (3D)
S_hat_X3 = np.kron(xs10, cxx3)
S_hat_Y3 = np.kron(ys10, cyy3)
S_hat_Z3 = np.kron(zs10, czz3)
# However there is only four terms that we should change the coin state when the jumping in space is happening (3D)
zx3_57_51 = np.kron(node57_51, czx3)
xz3_51_57 = np.kron(node51_57, cxz3)
xz3_60_55 = np.kron(node60_55, cxz3)
zx3_55_60 = np.kron(node55_60, czx3)
#=== Finally the Shift operator is:===
```

```
S_hatQ3 = S_hat_X3 + S_hat_Y3 + S_hat_Z3 + zx3_57_51 + xz3_51_57 + xz3_60_55 + zx3_55_60
S_hatC3 = xs10 + ys10 + zs10 + node57_51 + node51_57 + node60_55 + node55_60
#===== Evolution operator ==========================
UG3 = S_hatQ3.dot(np.kron(np.eye(N), G3))# Grover, quantum 3D
UC3= S_hatC3/3 # classical case 3D
#======Define the initial state Psi(zero)=====================================
psi_0_Q3 = np.kron((node[1]+node[2]+node[3]+node[4]+node[5]+node[6])/np.sqrt(6.),(coin3[0]+coin3[1]+coin3[2])/np.sqrt(3.))
psi_0_C3 = (node[1]+node[2]+node[3]+node[4]+node[5]+node[6])/(6.)
#-----------------------------------------------------------------------
yy= T+1
Arri_Prob_G3 = np.zeros(yy)
Arri_Prob_C3 = np.zeros(yy)
aG3 =0
aC3 =0
bG3 = np.zeros(yy)
bC3 = np.zeros(yy)
Norm_psi_t_G3 = np.zeros(yy)
Norm_psi_t_C3 = np.zeros(yy)
#========================================
t=0
while t<=T:
        if t==0:
                psi_t_G3 = psi_0_Q3
                psi_t_C3 = psi_0_C3
        else :
                psi_t_G3 = np.dot(UG3, psi_t_G3)
                psi_t_C3 = np.dot(UC3, psi_t_C3)
        #====Grover 3D================================
        e3 = ((54)*3)
        while e3<(60*3):
        psi_t_G3[e3] = 0
        e3+=1
        #---------------------------------------------
        Norm_psi_t_G3[t] = psi_t_G3.dot(psi_t_G3.conjugate()).real
        Arri_Prob_G3[t] = 1- Norm_psi_t_G3[t]
        #=========classical case 3D=====================
        psi_t_C3[54]=0
        psi_t_C3[55]=0
        psi_t_C3[56]=0
        psi_t_C3[57]=0
        psi_t_C3[58]=0
        psi_t_C3[59]=0
        #---------------------------------------------
        Norm_psi_t_C3[t] = np.sum(psi_t_C3)
        Arri_Prob_C3[t] = 1 - Norm_psi_t_C3[t]
        #-----------------------------------------------------------
        t+=1
t-=1
#====the plotting part =================
minorLocator = AutoMinorLocator()
fig, ax = plt.subplots()
plt.plot(Arri_Prob_G3, 'g-', label='G 3D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_C3, 'k-', label='C 3D', markersize=12, linewidth=7)
ax.xaxis.set_minor_locator(minorLocator)
ax.tick_params(axis='x', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='x', which='minor', labelsize=70, width=2, length=20, colors='k')
ax.tick_params(axis='y', which='major', labelsize=70, width=2, length=20, colors='k')
ax.set_xlabel('Steps number', fontsize=65)
ax.set_ylabel('Arrival probability', color='k', fontsize=65)
for tl in ax.get_yticklabels():
tl.set_color('k')
plt.xlim(0, yy)
plt.ylim(0, 1)
plt.title('Arrival probability,C06. 8 Levels, starting from 6 vertex (hexagonal) and equal mixture of all coin states .after '
+ str(T) + ' steps', fontsize=16)
plt.legend(fontsize=20, loc='best', borderaxespad=0.)
figManager = plt.get_current_fig_manager()
figManager.window.showMaximized()
plt.tight_layout()
plt.show()
print ('الحمد لله')
#--------end of the code-------------------------------------------------------
```

البرنامج 6: تطور الاحتمال الواصل لأنبوب ZigZag الحلقي

```
print ('بسم الله الرحمان الرحيم')
#author: Hamza Bougroura
#========= Import the requirement module ===================
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
#=======Define the critical number ========================
```

```
T = 600          # number of random steps
W = 6            # number of vertex in width direction (any number)
R = (2*8)+2      # the number of rings (should be even number)
Lev = (R/2)+1    # number of Level in loop
N = W*R          # the whole(entire) number of vertex
O=1
X=1
Y=1
Z=1
#===Define Grover coin operator 3D =================
G3 = (np.array([[-1, 2, 2], [2, -1, 2], [2, 2, -1]]))/(3.)
#=====generate coin state ===================================
coin3 = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
#=====generate space state that represent nodes ===============
node = np.zeros((N+1, N))
k=0
while k<N:
        space = np.zeros(N)
        space[k] = 1
        node[k+1] = space
        k+=1
#====generate steps matrix ==========================
L=N-W
r=v=i=1
os=xs=ys=zs= 0
while r<=R:
        a=b=c=d=1
        while v <= (W*r):
                if i==1:
                        if a==1:
                                a1=(2*W)-1
                                y = np.outer(node[v+a1], node[v])
                                ys = ys + y
                                a = a + 1
                        else:
                                a2=W-1
                                y = np.outer(node[v+a2], node[v])
                                ys = ys + y
                if i==2:
                        if b==W:
                                bw=(2*W)-1
                                y = np.outer(node[v-bw], node[v])
                                ys = ys + y
                        else:
                                b2=W-1
                                y = np.outer(node[v-b2], node[v])
                                ys = ys + y
                                b+=1
                if i==3:
                        if c==W:
                                y = np.outer(node[v+1], node[v])
                                ys = ys + y
                        else:
                                c2=(W+1)
                                y = np.outer(node[v+c2], node[v])
                                ys = ys + y
                                c+=1
                if i==4:
                        if d==1:
                                y = np.outer(node[v-1], node[v])
                                ys = ys + y
                                d = d + 1
                        else:
                                d2=W+1
                                y = np.outer(node[v-d2], node[v])
                                ys = ys + y
                v+=1
        r+=1
        i+=1
        if i==5:
                i=1

r=v=1
while r<=R:
        while v<=(W*r):
                if r==1:
                        z = np.outer(node[v+W], node[v])
                        x = np.outer(node[v+L], node[v])
                        o = np.outer(node[v], node[v])
                        os = os + o
                        zs = zs + z
                        xs = xs + x
                else:
```

```
                        if r==R:
                                z = np.outer(node[v-L], node[v])
                                x = np.outer(node[v-W], node[v])
                                o = np.outer(node[v], node[v])
                                os = os + o
                                zs = zs + z
                                xs = xs + x
                        else:
                                z = np.outer(node[v+W], node[v])
                                x = np.outer(node[v-W], node[v])
                                o = np.outer(node[v], node[v])
                                os = os + o
                                zs = zs + z
                                xs = xs + x
                v+=1
        r+=1

# the coin state terms that we need without change (in 3D)
cxx3 = np.outer(coin3[0], coin3[0])
cyy3 = np.outer(coin3[1], coin3[1])
czz3 = np.outer(coin3[2], coin3[2])
# the coin state terms that we need with change (in 3D)
czx3 = np.outer(coin3[2], coin3[0])
cxz3 = np.outer(coin3[0], coin3[2])
# in the following terms no change in the coin state when the jumping in space is happening (3D)
S_hat_X3 = np.kron(xs, czx3)
S_hat_Y3 = np.kron(ys, cyy3)
S_hat_Z3 = np.kron(zs, cxz3)
# Finally the Shift operator is:
S_hatQ3 = S_hat_X3 + S_hat_Y3 + S_hat_Z3
S_hatC3 = xs + ys + zs
print ('these results are for W=' + str(W) + ', and R =' + str(R) + '.')
#======Evolution operator =============
UG3 = S_hatQ3.dot(np.kron(np.eye(N), G3))
UC3 = S_hatC3/3
#======Difine the initial state Psi(zero)=====
i=1
space = np.zeros(N)
while i<=W:
        space = space + node[i]
        i+=1
psi0Q3 = np.kron((space)/np.sqrt(W),((coin3[0]*X)+(coin3[1]*Y)+(coin3[2]*Z))/np.sqrt((X)+(Y)+(Z)))
psi0C3 = ((space))/(W)
#-----------------------------------------------------------------------------------------------
yy= T+1
Arri_Prob_G3 = np.zeros(yy)
Arri_Prob_C3 = np.zeros(yy)
aG3 =0
aC3 =0
bG3 = np.zeros(yy)
bC3 = np.zeros(yy)
Norm_psi_t_G3 = np.zeros(yy)
Norm_psi_t_C3 = np.zeros(yy)
#=============================================
t=0
while t<=T:
        if t==0:
                psi_t_G3 = psi0Q3
                psi_t_C3 = psi0C3
        else :
                psi_t_G3 = np.dot(UG3, psi_t_G3)
                psi_t_C3 = np.dot(UC3, psi_t_C3)
        #====Grover 3D================================
        e3 = ((Lev-1)*W*3)
        while e3<(Lev*W*3):
                psi_t_G3[e3] = 0
                e3+=1
        #---------------------------------------------
        Norm_psi_t_G3[t] = psi_t_G3.dot(psi_t_G3.conjugate()).real
        Arri_Prob_G3[t] = 1 - Norm_psi_t_G3[t]
        #=========classical case 3D=====================
        s = 0
        while s<W:
                psi_t_C3[((Lev-1)*W)+s] = 0
                s+=1
        #---------------------------------
        Norm_psi_t_C3[t]=np.sum(psi_t_C3)
        Arri_Prob_C3[t] = 1 - Norm_psi_t_C3[t]
        t+=1
t-=1
#===== the plotting part ==========================================
minorLocator = AutoMinorLocator()
fig, ax = plt.subplots()
```

```
plt.plot(Arri_Prob_G3, 'g-', label='G 3D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_C3, 'k-', label='C 3D', markersize=12, linewidth=7)
ax.xaxis.set_minor_locator(minorLocator)
ax.tick_params(axis='x', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='x', which='minor', labelsize=70, width=2, length=20, colors='k')
ax.tick_params(axis='y', which='major', labelsize=70, width=2, length=20, colors='k')
ax.set_xlabel('Steps number', fontsize=65)
ax.set_ylabel('Arrival probability', color='k', fontsize=65)
for tl in ax.get_yticklabels():
tl.set_color('k')
plt.xlim(0, yy)
plt.ylim(0, 1)
plt.title('Arrival probability.ZigZag loop ('+str(Lev)+' Levels),'+str(W)+' width,'+str(R)+' ring,initial state: '+str(W)+' node
+ equal mixture of all coin states.after '+str(T)+' steps', fontsize=16)
plt.legend(fontsize=20, loc='best', borderaxespad=0.)
figManager = plt.get_current_fig_manager()
figManager.window.showMaximized()
plt.tight_layout()
plt.show()
print ('الحمد لله')
#------the End of the code ----------------------------------------------------------------------------------------
```

البرنامج 7: تطور الاحتمال الواصل لأنبوب ArmChair الحلقي

```
print ('بسم الله الرحمان الرحيم')
#author: Hamza Bougroura
#====== Import the requirement module ======================
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
#======== Define the critical number ======================
T = 1000       # number of random steps
W = 6          # number of vertex in width direction (any number)
R = (2*8)+2    # the number of rings (should be even number)
Lev = (R/2)+1  # number of Level in loop
N = W*R        # the whole(entire) number of vertex
O=1
X=1
Y=1
Z=1
#===Define Grover coin operator 3D =================
G3 = (np.array([[-1, 2, 2], [2, -1, 2], [2, 2, -1]]))/(3.)
#=====generate coin state ===================================
coin3 = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
#=====generate space state that represent nodes ==============
node = np.zeros((N+1, N))
k=0
while k<N:
        space = np.zeros(N)
        space[k] = 1
        node[k+1] = space
        k+=1
#====generate steps matrix ==========================
r=v= 1
os=xs=ys=zs= 0
while r<=R:
        i=1
        while v <= (W*r):
                if r%2==0:
                        if i==1:
                                ye1 = np.outer(node[v+(W-1)], node[v])
                                ys = ys + ye1
                                i = i + 1
                        else:
                                if i==W:
                                        yew = np.outer(node[v-(W-1)], node[v])
                                        ys = ys + yew
                                else:
                                        if v%2==0:
                                                yee = np.outer(node[v+1], node[v])
                                                ys = ys + yee
                                                i = i + 1
                                        else:
                                                yeo = np.outer(node[v-1], node[v])
                                                ys = ys + yeo
                                                i = i + 1
                else:
                        if v%2==0:
                                yoe = np.outer(node[v-1], node[v])
                                ys = ys + yoe
                        else:
                                yoo = np.outer(node[v+1], node[v])
```

```
                                ys = ys + yoo
                v+=1
        r+=1

r=1
v=1
L=N-W
while r<=R:
        while v<=(W*r):
                if r==1:
                        z = np.outer(node[v+W], node[v])
                        x = np.outer(node[v+L], node[v])
                        o = np.outer(node[v], node[v])
                        os = os + o
                        xs = xs + x
                        zs = zs + z
                else:
                        if r==R:
                                z = np.outer(node[v-L], node[v])
                                x = np.outer(node[v-W], node[v])
                                o = np.outer(node[v], node[v])
                                os = os + o
                                xs = xs + x
                                zs = zs + z
                        else:
                                z = np.outer(node[v+W], node[v])
                                x = np.outer(node[v-W], node[v])
                                o = np.outer(node[v], node[v])
                                os = os + o
                                xs = xs + x
                                zs = zs + z
                v+=1
        r+=1
# the coin state terms that we need without change (in 3D)
cxx3 = np.outer(coin3[0], coin3[0])
cyy3 = np.outer(coin3[1], coin3[1])
czz3 = np.outer(coin3[2], coin3[2])
# the coin state terms that we need with change (in 3D)
czx3 = np.outer(coin3[2], coin3[0])
cxz3 = np.outer(coin3[0], coin3[2])
# in the following terms not change in the coin state when the jumping in space is happening (3D)
S_hat_X3 = np.kron(xs, czx3)
S_hat_Y3 = np.kron(ys, cyy3)
S_hat_Z3 = np.kron(zs, cxz3)
# Finally the Shift operator is:
S_hatQ3 = S_hat_X3 + S_hat_Y3 + S_hat_Z3
S_hatC3 = xs + ys + zs
print ('these results are for W=' + str(W) + ', and R =' + str(R) + '.')
#======Evolution operator =============
UG3 = S_hatQ3.dot(np.kron(np.eye(N), G3))
UC3= S_hatC3/3
#======Difine the initial state Psi(zero)=====
i=1
space = np.zeros(N)
while i<=W:
        space = space + node[i]
        i+=1
psi0Q3 = np.kron((space)/np.sqrt(W),((coin3[0]*X)+(coin3[1]*Y)+(coin3[2]*Z))/np.sqrt((X)+(Y)+(Z)))
psi0C3 = ((space))/(W)
#-------------------------------------------------------------------------------------------------------------------
yy= T+1
Arri_Prob_G3 = np.zeros(yy)
Arri_Prob_C3 = np.zeros(yy)
aG3 =0
aC3 =0
bG3 = np.zeros(yy)
bC3 = np.zeros(yy)
Norm_psi_t_G3 = np.zeros(yy)
Norm_psi_t_C3 = np.zeros(yy)
#============================================
t=0
while t<=T:
        if t==0:
                psi_t_G3 = psi0Q3
                psi_t_C3 = psi0C3
        else :
                psi_t_G3 = np.dot(UG3, psi_t_G3)
                psi_t_C3 = np.dot(UC3, psi_t_C3)
        #====Grover 3D=================================
        e3 = ((Lev-1)*W*3)
        while e3<(Lev*W*3):
        psi_t_G3[e3] = 0
        e3+=1
```

```
        #-------------------------------------------
        Norm_psi_t_G3[t] = psi_t_G3.dot(psi_t_G3.conjugate()).real
        Arri_Prob_G3[t] = 1 - Norm_psi_t_G3[t]
        #=========classical case 3D======================
        s = 0
        while s<W:
        psi_t_C3[((Lev-1)*W)+s] = 0
        s+=1
        #---------------------------------
        Norm_psi_t_C3[t]=np.sum(psi_t_C3)
        Arri_Prob_C3[t] = 1 - Norm_psi_t_C3[t]
        t+=1
t-=1
#===== the plotting part ==========================================
minorLocator = AutoMinorLocator()
fig, ax = plt.subplots()
plt.plot(Arri_Prob_G3, 'g-', label='G 3D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_C3, 'k-', label='C 3D', markersize=12, linewidth=7)
ax.xaxis.set_minor_locator(minorLocator)
ax.tick_params(axis='x', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='x', which='minor', labelsize=70, width=2, length=20, colors='k')
ax.tick_params(axis='y', which='major', labelsize=70, width=2, length=20, colors='k')
ax.set_xlabel('Steps number', fontsize=65)
ax.set_ylabel('Arrival probability', color='k', fontsize=65)
for tl in ax.get_yticklabels():
tl.set_color('k')
plt.xlim(0, yy)
plt.ylim(0, 1)
plt.title('Transport probability.ArmChair loop ('+str(Lev)+' Levels),'+str(W)+' width,'+str(R)+' ring,initial state: '+str(W)+'
node + equal mixture of all coin states.after '+str(T)+' steps', fontsize=16)
plt.legend(fontsize=20, loc='best', borderaxespad=0.)
figManager = plt.get_current_fig_manager()
figManager.window.showMaximized()
plt.tight_layout()
plt.show()
print ('الحمد لله')
#------the End of the code ---------------------------------------------------------------------------------------
```

البرنامج 8: تطور الاحتمال الواصل لأنبوب ZigZag المقبب

```
print ('بسم الله الرحمان الرحيم')
#author: Hamza Bougroura
#======== Import the requirement module ===================
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
#======= Define the critical number ======================
T = 6000              # number of random steps
aa = 2                # this aa number because the number of the ring muset be even
R = aa*16             # number of ring (only in the open tube)
W = 9                 # the width of each ring
Lev = R + 8           # the Levels number of the whole capped tube
N = (W*R)+60          # number of entier of vertex in space (vertex)
#===Define Grover coin operator 3D =================
G3 = (np.array([[-1, 2, 2], [2, -1, 2], [2, 2, -1]]))/(3.)
#=====generate coin state ===================================
coin3 = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
#=====generate space state that represent nodes ===============
node = np.zeros((N+1, N))
k=0
while k<N:
        space = np.zeros(N)
        space[k] = 1
        node[k+1] = space
        k+=1
#====generate steps matrix ==================================
r=4
rr=R+(r-1)
v=31
i=4
os=xs=ys=zs= 0
xs_1 = 0
while r<=(rr):
        a=b=c=d=1
        ll=v
        while v <(W+ll):
                if i==1:
                        if a==1:
                                a1=(2*W)-1
                                y = np.outer(node[v+a1], node[v])
                                ys = ys + y
                                a = a + 1
```

```
                        else:
                                a2=W-1
                                y = np.outer(node[v+a2], node[v])
                                ys = ys + y
                if i==2:
                        if b==W:
                                bw=(2*W)-1
                                y = np.outer(node[v-bw], node[v])
                                ys = ys + y
                        else:
                                b2=W-1
                                y = np.outer(node[v-b2], node[v])
                                ys = ys + y
                                b+=1
                if i==3:
                        if c==W:
                                y = np.outer(node[v+1], node[v])
                                ys = ys + y
                        else:
                                c2=(W+1)
                                y = np.outer(node[v+c2], node[v])
                                ys = ys + y
                                c+=1
                if i==4:
                        if d==1:
                                y = np.outer(node[v-1], node[v])
                                ys = ys + y
                                d = d + 1
                        else:
                                d2=W+1
                                y = np.outer(node[v-d2], node[v])
                                ys = ys + y
                v+=1
        r+=1
        i+=1
        if i==5:
                i=1
r=4
rr=R+(r-1)
v=31
while r<=rr:
        ll=v
        while v<(W+ll):
                if r==4:
                        z = np.outer(node[v+W], node[v])
                        x = np.outer(node[v-W], node[v])
                        o = np.outer(node[v], node[v])
                        os = os + o
                        zs = zs + z
                        xs_1 = xs_1 + x
                else:
                        z = np.outer(node[v+W], node[v])
                        x = np.outer(node[v-W], node[v])
                        o = np.outer(node[v], node[v])
                        os = os + o
                        zs = zs + z
                        xs = xs + x
                v+=1
        r+=1
v-=1
v = v - 30
#====generate the space terms For the two half of the C60 sphere alone starting from 6 vertex case
(Hexagonal)======================
#|O><O|
os1 = np.outer(node[1], node[1])+np.outer(node[2], node[2]) +np.outer(node[3], node[3])+np.outer(node[4],
node[4])+np.outer(node[5], node[5])+np.outer(node[6], node[6])+np.outer(node[7], node[7])+np.outer(node[8],
node[8])+np.outer(node[9], node[9])+np.outer(node[10], node[10])
os2 = os1 +np.outer(node[11], node[11])+np.outer(node[12], node[12])+np.outer(node[13], node[13])+np.outer(node[14],
node[14])+np.outer(node[15], node[15])+np.outer(node[16], node[16])+np.outer(node[17], node[17])+np.outer(node[18],
node[18])+np.outer(node[19], node[19])+np.outer(node[20], node[20])
os3 = os2 +np.outer(node[21], node[21])+np.outer(node[22], node[22])+np.outer(node[23], node[23])+np.outer(node[24],
node[24])+np.outer(node[25], node[25])+np.outer(node[26], node[26])+np.outer(node[27], node[27])+np.outer(node[28],
node[28])+np.outer(node[29], node[29])+np.outer(node[30], node[30])
#-------------------------------------------------------------------------------------------------------------------------------
os4 = os3 +np.outer(node[v+31], node[v+31])+np.outer(node[v+32], node[v+32])+np.outer(node[v+33],
node[v+33])+np.outer(node[v+34], node[v+34])+np.outer(node[v+35], node[v+35])+np.outer(node[v+36],
node[v+36])+np.outer(node[v+37], node[v+37])+np.outer(node[v+38], node[v+38])+np.outer(node[v+39],
node[v+39])+np.outer(node[v+40], node[v+40])
os5 = os4 +np.outer(node[v+41], node[v+41])+np.outer(node[v+42], node[v+42])+np.outer(node[v+43],
node[v+43])+np.outer(node[v+44], node[v+44])+np.outer(node[v+45], node[v+45])+np.outer(node[v+46],
node[v+46])+np.outer(node[v+47], node[v+47])+np.outer(node[v+48], node[v+48])+np.outer(node[v+49],
node[v+49])+np.outer(node[v+50], node[v+50])
```

```
os6 = os5 +np.outer(node[v+51], node[v+51])+np.outer(node[v+52], node[v+52])+np.outer(node[v+53],
node[v+53])+np.outer(node[v+54], node[v+54])+np.outer(node[v+55], node[v+55])+np.outer(node[v+56],
node[v+56])+np.outer(node[v+57], node[v+57])+np.outer(node[v+58], node[v+58])+np.outer(node[v+59],
node[v+59])+np.outer(node[v+60], node[v+60])

#|X><X|: don't forget to put back the term "np.outer(node[v+49], node[v+40])" in its right place
xs1 = np.outer(node[6], node[1]) +np.outer(node[3], node[2]) +np.outer(node[2], node[3]) +np.outer(node[5], node[4])
+np.outer(node[4], node[5]) +np.outer(node[1], node[6]) +np.outer(node[14], node[7]) +np.outer(node[15], node[8])
+np.outer(node[17], node[9]) +np.outer(node[18], node[10])+np.outer(node[v+49], node[v+40])
xs2 = xs1 +np.outer(node[20], node[11])+np.outer(node[21], node[12])+ +np.outer(node[7], node[14]) +np.outer(node[8],
node[15]) + np.outer(node[9], node[17])+np.outer(node[10], node[18])+ +np.outer(node[11], node[20])
xs3 = xs2 +np.outer(node[12], node[21])+np.outer(node[31], node[22])+np.outer(node[32], node[23])+np.outer(node[33],
node[24])+np.outer(node[34], node[25])+np.outer(node[35], node[26])+np.outer(node[36], node[27])+np.outer(node[37],
node[28])+np.outer(node[38], node[29])+np.outer(node[39], node[30])
#-------------------------------------------------------------------------------------------------------------------
xs4= np.outer(node[v+22], node[v+31])+np.outer(node[v+23], node[v+32])+np.outer(node[v+24], node[v+33])+np.outer(node[v+25],
node[v+34])+np.outer(node[v+26], node[v+35])+np.outer(node[v+27], node[v+36])+np.outer(node[v+28],
node[v+37])+np.outer(node[v+29], node[v+38])+np.outer(node[v+30], node[v+39])
xs5 = xs3+ +np.outer(node[v+51], node[v+42])+ + +np.outer(node[v+53], node[v+45])+ +np.outer(node[v+54], node[v+47])+
+np.outer(node[v+40], node[v+49])
xs6 = xs5+np.outer(node[v+42], node[v+51])+np.outer(node[v+58], node[v+52])+np.outer(node[v+45],
node[v+53])+np.outer(node[v+47], node[v+54])+np.outer(node[v+60], node[v+55])+np.outer(node[v+57],
node[v+56])+np.outer(node[v+56], node[v+57])+np.outer(node[v+52], node[v+58])+ + np.outer(node[v+55], node[v+60])

#|Y><Y|
ys1 = + +np.outer(node[8], node[2]) +np.outer(node[4], node[3]) +np.outer(node[3], node[4]) +np.outer(node[11], node[5])
+np.outer(node[12], node[6]) +np.outer(node[13], node[7]) +np.outer(node[2], node[8]) +np.outer(node[16], node[9])
+np.outer(node[19], node[10])
ys2 = ys1 +np.outer(node[5], node[11]) +np.outer(node[6], node[12]) +np.outer(node[7], node[13]) +np.outer(node[15],
node[14])+np.outer(node[14], node[15])+np.outer(node[9], node[16]) +np.outer(node[18], node[17])+np.outer(node[17],
node[18])+np.outer(node[10], node[19])+np.outer(node[21], node[20])
ys3 = ys2 +np.outer(node[20], node[21])+np.outer(node[32], node[22])+np.outer(node[33], node[23])+np.outer(node[34],
node[24])+np.outer(node[35], node[25])+np.outer(node[36], node[26])+np.outer(node[37], node[27])+np.outer(node[38],
node[28])+np.outer(node[39], node[29])+np.outer(node[31], node[30])
#-------------------------------------------------------------------------------------------------------------------
ys4 = ys3 +np.outer(node[v+30], node[v+31])+np.outer(node[v+22], node[v+32])+np.outer(node[v+23],
node[v+33])+np.outer(node[v+24], node[v+34])+np.outer(node[v+25], node[v+35])+np.outer(node[v+26],
node[v+36])+np.outer(node[v+27], node[v+37])+np.outer(node[v+28], node[v+38])+np.outer(node[v+29],
node[v+39])+np.outer(node[v+41], node[v+40])
ys5 = ys4 +np.outer(node[v+40], node[v+41])+np.outer(node[v+50], node[v+42])+np.outer(node[v+44],
node[v+43])+np.outer(node[v+43], node[v+44])+np.outer(node[v+52], node[v+45])+np.outer(node[v+47],
node[v+46])+np.outer(node[v+46], node[v+47])+np.outer(node[v+49], node[v+48])+np.outer(node[v+48],
node[v+49])+np.outer(node[v+42], node[v+50])
ys6 = ys5 + +np.outer(node[v+45], node[v+52])+np.outer(node[v+59], node[v+53])+np.outer(node[v+60],
node[v+54])+np.outer(node[v+56], node[v+55])+np.outer(node[v+55], node[v+56])+np.outer(node[v+58],
node[v+57])+np.outer(node[v+57], node[v+58])+np.outer(node[v+53], node[v+59])+np.outer(node[v+54], node[v+60])

#|Z><Z|
zs1 = np.outer(node[7], node[1])+ +np.outer(node[9], node[3])+np.outer(node[10], node[4]) +np.outer(node[6], node[5])
+np.outer(node[5], node[6]) +np.outer(node[1], node[7]) + +np.outer(node[3], node[9]) +np.outer(node[4], node[10])
zs2 = zs1 + + +np.outer(node[22], node[13])+np.outer(node[23], node[14])+np.outer(node[24], node[15])+np.outer(node[25],
node[16])+np.outer(node[26], node[17])+np.outer(node[27], node[18])+np.outer(node[28], node[19])+np.outer(node[29], node[20])
zs3 = zs2 +np.outer(node[30], node[21])+np.outer(node[13], node[22])+np.outer(node[14], node[23])+np.outer(node[15],
node[24])+np.outer(node[16], node[25])+np.outer(node[17], node[26])+np.outer(node[18], node[27])+np.outer(node[19],
node[28])+np.outer(node[20], node[29])+np.outer(node[21], node[30])
#-------------------------------------------------------------------------------------------------------------------
zs4 = zs3 +np.outer(node[v+40], node[v+31])+np.outer(node[v+41], node[v+32])+np.outer(node[v+42],
node[v+33])+np.outer(node[v+43], node[v+34])+np.outer(node[v+44], node[v+35])+np.outer(node[v+45],
node[v+36])+np.outer(node[v+46], node[v+37])+np.outer(node[v+47], node[v+38])+np.outer(node[v+48],
node[v+39])+np.outer(node[v+31], node[v+40])
zs5 = zs4 +np.outer(node[v+32], node[v+41])+np.outer(node[v+33], node[v+42])+np.outer(node[v+34],
node[v+43])+np.outer(node[v+35], node[v+44])+np.outer(node[v+36], node[v+45])+np.outer(node[v+37],
node[v+46])+np.outer(node[v+38], node[v+47])+np.outer(node[v+39], node[v+48])+np.outer(node[v+55], node[v+49])
zs6 = zs5 +np.outer(node[v+57], node[v+51])+ + + +np.outer(node[v+49], node[v+55])+ +np.outer(node[v+51], node[v+57])+
+np.outer(node[v+60], node[v+59])+np.outer(node[v+59], node[v+60])

# the except four jumping space are(corresponding to |Z><X| and |X><Z|)
node_zy = np.outer(node[2], node[1])
node_yz = np.outer(node[1], node[2])

node_zx = np.outer(node[8], node[16])+np.outer(node[11], node[19])+np.outer(node[12], node[13])+np.outer(node[v+50],
node[v+41])+np.outer(node[v+52], node[v+44])+np.outer(node[v+53], node[v+46])+np.outer(node[v+54],
node[v+48])+np.outer(node[v+56], node[v+50])+np.outer(node[v+58], node[v+59])

node_xz = np.outer(node[16], node[8])+np.outer(node[19], node[11])+np.outer(node[13], node[12])+np.outer(node[v+41],
node[v+50])+np.outer(node[v+44], node[v+52])+np.outer(node[v+46], node[v+53])+np.outer(node[v+48],
node[v+54])+np.outer(node[v+50], node[v+56])+np.outer(node[v+59], node[v+58])

node_yx = np.outer(node[v+51], node[v+43])
node_xy = np.outer(node[v+43], node[v+51])
#-----------------------------------------------------------
## the coin state terms that we need without change (in 3D)
cxx3 = np.outer(coin3[0], coin3[0])
```

```
cyy3 = np.outer(coin3[1], coin3[1])
czz3 = np.outer(coin3[2], coin3[2])
# the coin state terms that we need with change (in 3D)
czx3 = np.outer(coin3[2], coin3[0])
cxz3 = np.outer(coin3[0], coin3[2])
#----
cyx3 = np.outer(coin3[1], coin3[0])
cxy3 = np.outer(coin3[0], coin3[1])
#-----
czy3 = np.outer(coin3[2], coin3[1])
cyz3 = np.outer(coin3[1], coin3[2])
#----------------------------------------
osT = os6 + os
xsT = xs6 + xs_1
ysT = ys6 + ys
zsT = zs6
#----------------------------------------
# in the following terms (of the Shift operator) there is not change in the coin state when the jumping in space is happening
(3D)
S_hat_X3 = np.kron(xsT, cxx3)
S_hat_Y3 = np.kron(ysT, cyy3)
S_hat_Z3 = np.kron(zsT, czz3)
# However there is only four terms (of the Shift operator) that we should change the coin state when the jumping in space is
happening (3D)
S_hat_zx3 = np.kron(node_zx, czx3)+np.kron(xs4, czx3)+np.kron(xs, czx3)
S_hat_xz3 = np.kron(node_xz, cxz3)+np.kron(zs, cxz3)
#-----
S_hat_yx3 = np.kron(node_yx, cyx3)
S_hat_xy3 = np.kron(node_xy, cxy3)
#-----
S_hat_zy3 = np.kron(node_zy, czy3)
S_hat_yz3 = np.kron(node_yz, cyz3)
#------------------------------------------------------------
# Finally the Shift operator is:
S_hatQ3 = S_hat_X3 + S_hat_Y3 + S_hat_Z3 + S_hat_zx3 + S_hat_xz3 + S_hat_yx3 + S_hat_xy3 + S_hat_zy3 + S_hat_yz3
S_hatC3 = xsT + ysT + zsT + node_xz + node_zx + node_xy + node_yx + node_yz + node_zy + xs4 + xs +zs
#----------------------------------------------
invers_S_hatQ3 = np.linalg.inv(S_hatQ3)
unitarity_of_S_hatQ3 = np.dot(invers_S_hatQ3, S_hatQ3)
print ('unitarity of S_hatQ3 = ')
print (unitarity_of_S_hatQ3)

print ('these results are for W=' + str(W) + ', and R =' + str(R) + '.')
print ('++++++++++++++++++++++++++++')
#===== Evolution operator =========
UG3 = S_hatQ3.dot(np.kron(np.eye(N), G3))
UC3= S_hatC3/3
#======Difine the initial state Psi( zero)=====================
i=1
space = np.zeros(N)
while i<=(6):
space = space + node[i]
i+=1
psi0Q3 = np.kron((space)/np.sqrt(6),(coin3[0]+coin3[1]+coin3[2])/np.sqrt(3.))
psi0C3 =((space))/(6)
#=============================================
yy= T+1
Arri_Prob_G3 = np.zeros(yy)
Arri_Prob_C3 = np.zeros(yy)
aG3 =0
aC3 =0
bG3 = np.zeros(yy)
bC3 = np.zeros(yy)
Norm_psi_t_G3 = np.zeros(yy)
Norm_psi_t_C3 = np.zeros(yy)
#=============================================
t=0
while t<=T:
        if t==0:
                psi_t_G3 = psi0Q3
                psi_t_C3 = psi0C3
        else :
                psi_t_G3 = np.dot(UG3, psi_t_G3)
                psi_t_C3 = np.dot(UC3, psi_t_C3)
        #====Grover 3D================================
        e3 = ((v+54)*3)
        while e3<((v+60)*3):
                psi_t_G3[e3] = 0
                e3+=1
        #---------------------------------------------
        Norm_psi_t_G3[t] = psi_t_G3.dot(psi_t_G3.conjugate()).real
        Arri_Prob_G3[t] = 1 - Norm_psi_t_G3[t]
        #=========classical case 3D======================
```

```
        psi_t_C3[v+54]=0
        psi_t_C3[v+55]=0
        psi_t_C3[v+56]=0
        psi_t_C3[v+57]=0
        psi_t_C3[v+58]=0
        psi_t_C3[v+59]=0
        #---------------------------------
        Norm_psi_t_C3[t]=np.sum(psi_t_C3)
        Arri_Prob_C3[t] = 1 - Norm_psi_t_C3[t]
        t+=1
t-=1
#===== the plotting part ==========================================
minorLocator = AutoMinorLocator()
fig, ax = plt.subplots()
plt.plot(Arri_Prob_G3, 'g-', label='G 3D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_C3, 'k-', label='C 3D', markersize=12, linewidth=7)
ax.xaxis.set_minor_locator(minorLocator)
ax.tick_params(axis='x', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='x', which='minor', labelsize=70, width=2, length=20, colors='k')
ax.tick_params(axis='y', which='major', labelsize=70, width=2, length=20, colors='k')
ax.set_xlabel('Steps number', fontsize=65)
ax.set_ylabel('Arrival probability', color='k', fontsize=65)
for tl in ax.get_yticklabels():
tl.set_color('k')
plt.xlim(0, yy)
plt.ylim(0, 1)
plt.title('Arrival probability.ZigZag capped ('+str(Lev)+' Levels),'+str(W)+' width,'+str(R)+'+8 ring, initial state: 6
node(Hexagonal) + equal mixture of all coin states.after '+str(T)+' steps', fontsize=16)
plt.legend(fontsize=20, loc='best', borderaxespad=0.)
figManager = plt.get_current_fig_manager()
figManager.window.showMaximized()
plt.tight_layout()
plt.show()
print ('الحمد لله')
#------the End of the code ---------------------------------------------------------------------------------------
```

البرنامج 9: تطور الاحتمال الواصل لأنبوب ArmChair المقبب

```
print ('بسم الله الرحمان الرحيم')
#author: Hamza Bougroura
#======== Import the requirement module ===================
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
#======= Define the critical number ======================
T = 60000       # number of random steps
R = 32          # number of ring only in the open tube
W = 10          # the width of each ring
Lev = R + 8     #the Levels number of the whole closed tube
N = (W*R)+60    # number of positions in space (vertex)
#===Define Grover coin operator 3D =================
G3 = (np.array([[-1, 2, 2], [2, -1, 2], [2, 2, -1]]))/(3.)
#=====generate coin state ===================================
coin3 = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
#=====generate space state that represent nodes ===============
node = np.zeros((N+1, N))
k=0
while k<N:
        space = np.zeros(N)
        space[k] = 1
        node[k+1] = space
        k+=1
#====generate steps matrix ==================================
r=4
v=31
os_P2=0
xs_P2_xx=0
xs_P2=0
ys_P2=0
zs_P2= 0
while r<=(R+3):
        i=1
        while v <= (W*r):
                if r%2==0:
                        if i==1:
                                ye1 = np.outer(node[v+(W-1)], node[v])
                                ys_P2 = ys_P2 + ye1
                                i = i + 1
                        else:
                                if i==W:
                                        yew = np.outer(node[v-(W-1)], node[v])
                                        ys_P2 = ys_P2 + yew
```

```
                                else:
                                        if v%2==0:
                                                yee = np.outer(node[v+1], node[v])
                                                ys_P2 = ys_P2 + yee
                                                i = i + 1
                                        else:
                                                yeo = np.outer(node[v-1], node[v])
                                                ys_P2 = ys_P2 + yeo
                                                i = i + 1
                        else:
                                if v%2==0:
                                        yoe = np.outer(node[v-1], node[v])
                                        ys_P2 = ys_P2 + yoe
                                else:
                                        yoo = np.outer(node[v+1], node[v])
                                        ys_P2 = ys_P2 + yoo
                        v+=1
                r+=1
r=4
v=31
while r<=(R+3):
        while v<=(W*r):
                if r==4:
                        z = np.outer(node[v+W], node[v])
                        x = np.outer(node[v-W], node[v])
                        o = np.outer(node[v], node[v])
                        os_P2 = os_P2 + o
                        xs_P2_xx = xs_P2_xx + x
                        zs_P2 = zs_P2 + z
                else:
                        z = np.outer(node[v+W], node[v])
                        x = np.outer(node[v-W], node[v])
                        o = np.outer(node[v], node[v])
                        os_P2 = os_P2 + o
                        xs_P2 = xs_P2 + x
                        zs_P2 = zs_P2 + z
                v+=1
        r+=1
v-=1
v = v - 30
#-----------------------------------
#|O><O|
os1 = +np.outer(node[1], node[1])+np.outer(node[2], node[2])+np.outer(node[3], node[3])+np.outer(node[4],
node[4])+np.outer(node[5], node[5])+np.outer(node[6], node[6]) +np.outer(node[7], node[7]) +np.outer(node[8], node[8])
+np.outer(node[9], node[9]) +np.outer(node[10], node[10])
os2 = os1 +np.outer(node[11], node[11])+np.outer(node[12], node[12])+np.outer(node[13], node[13])+np.outer(node[14],
node[14])+np.outer(node[15], node[15])+np.outer(node[16], node[16])+np.outer(node[17], node[17])+np.outer(node[18],
node[18])+np.outer(node[19], node[19])+np.outer(node[20], node[20])
os_P1 = os2 +np.outer(node[21], node[21])+np.outer(node[22], node[22])+np.outer(node[23], node[23])+np.outer(node[24],
node[24])+np.outer(node[25], node[25])+np.outer(node[26], node[26])+np.outer(node[27], node[27])+np.outer(node[28],
node[28])+np.outer(node[29], node[29])+np.outer(node[30], node[30])
#-------------------------------------------------------------------------------------------------------------------------
os4=np.outer(node[v+31], node[v+31])+np.outer(node[v+32], node[v+32])+np.outer(node[v+33], node[v+33])+np.outer(node[v+34],
node[v+34])+np.outer(node[v+35], node[v+35])+np.outer(node[v+36], node[v+36])+np.outer(node[v+37],
node[v+37])+np.outer(node[v+38], node[v+38])+np.outer(node[v+39], node[v+39])+np.outer(node[v+40], node[v+40])
os5=os4 +np.outer(node[v+41], node[v+41])+np.outer(node[v+42], node[v+42])+np.outer(node[v+43],
node[v+43])+np.outer(node[v+44], node[v+44])+np.outer(node[v+45], node[v+45])+np.outer(node[v+46],
node[v+46])+np.outer(node[v+47], node[v+47])+np.outer(node[v+48], node[v+48])+np.outer(node[v+49],
node[v+49])+np.outer(node[v+50], node[v+50])
os_P3=os5 +np.outer(node[v+51], node[v+51])+np.outer(node[v+52], node[v+52])+np.outer(node[v+53],
node[v+53])+np.outer(node[v+54], node[v+54])+np.outer(node[v+55], node[v+55])+np.outer(node[v+56],
node[v+56])+np.outer(node[v+57], node[v+57])+np.outer(node[v+58], node[v+58])+np.outer(node[v+59],
node[v+59])+np.outer(node[v+60], node[v+60])

#|X><X|
xs1 = np.outer(node[2], node[1])+np.outer(node[1], node[2])+np.outer(node[4], node[3])+np.outer(node[3],
node[4])+np.outer(node[10], node[5]) +np.outer(node[11], node[6]) +np.outer(node[12], node[7]) +np.outer(node[15],
node[8])+np.outer(node[16], node[9])+np.outer (node[5], node[10])
xs2 = xs1 +np.outer(node[6], node[11])+np.outer(node[7], node[12])+np.outer(node[14], node[13])+np.outer(node[13],
node[14])+np.outer(node[8], node[15])+np.outer(node[9], node[16])+np.outer(node[20], node[19])+np.outer(node[19], node[20])
xs_P1 = xs2 +np.outer(node[31], node[21])+np.outer(node[32], node[22])+np.outer(node[33], node[23])+np.outer(node[34],
node[24])+np.outer(node[35], node[25])+np.outer(node[36], node[26])+np.outer(node[37], node[27])+np.outer(node[38],
node[28])+np.outer(node[39], node[29])+np.outer(node[40], node[30])
#-------------------------------------------------------------------------------------------------------------------------
xs_P3_zx= np.outer(node[v+21], node[v+31])+np.outer(node[v+22], node[v+32])+np.outer(node[v+23],
node[v+33])+np.outer(node[v+24], node[v+34])+np.outer(node[v+25], node[v+35])+np.outer(node[v+26],
node[v+36])+np.outer(node[v+27], node[v+37])+np.outer(node[v+28], node[v+38])+np.outer(node[v+29],
node[v+39])+np.outer(node[v+30], node[v+40])
xs5= np.outer(node[v+50], node[v+41])+np.outer(node[v+52], node[v+42])+np.outer(node[v+51], node[v+43])+np.outer(node[v+45],
node[v+44])+np.outer(node[v+44], node[v+45])+np.outer(node[v+55], node[v+46])+np.outer(node[v+54],
node[v+47])+np.outer(node[v+53], node[v+49])+np.outer(node[v+41], node[v+50])
```

```
xs_P3= xs5+np.outer(node[v+43], node[v+51])+np.outer(node[v+42], node[v+52])+np.outer(node[v+49],
node[v+53])+np.outer(node[v+47], node[v+54])+np.outer(node[v+46], node[v+55])+np.outer(node[v+57],
node[v+56])+np.outer(node[v+56], node[v+57])+np.outer(node[v+59], node[v+58])+np.outer(node[v+58], node[v+59])

#|Y><Y|
ys1 = np.outer(node[5], node[1])+np.outer (node[3], node[2])+np.outer (node[2], node[3])+np.outer (node[9], node[4])
+np.outer(node[1], node[5])+np.outer(node[20], node[6]) +np.outer(node[13], node[7])+np.outer(node[14], node[8])
+np.outer(node[4], node[9]) +np.outer(node[19], node[10])
ys2 = ys1 +np.outer(node[12], node[11])+np.outer(node[11], node[12])+np.outer(node[7], node[13])+np.outer(node[8],
node[14])+np.outer(node[16], node[15])+np.outer(node[15], node[16])+np.outer(node[18], node[17])+np.outer(node[17],
node[18])+np.outer(node[10], node[19])+np.outer(node[6], node[20])
ys_P1 = ys2 +np.outer(node[30], node[21])+np.outer(node[23], node[22])+np.outer(node[22], node[23])+np.outer(node[25],
node[24])+np.outer(node[24], node[25])+np.outer(node[27], node[26])+np.outer(node[26], node[27])+np.outer(node[29],
node[28])+np.outer(node[28], node[29])+np.outer(node[21], node[30])
#-------------------------------------------------------------------------------------------------------------------
ys4=np.outer(node[v+32], node[v+31])+np.outer(node[v+31], node[v+32])+np.outer(node[v+34], node[v+33])+np.outer(node[v+33],
node[v+34])+np.outer(node[v+36], node[v+35])+np.outer(node[v+35], node[v+36])+np.outer(node[v+38],
node[v+37])+np.outer(node[v+37], node[v+38])+np.outer(node[v+40], node[v+39])+np.outer(node[v+39], node[v+40])
ys5=ys4 +np.outer(node[v+52], node[v+41])+np.outer(node[v+43], node[v+42])+np.outer(node[v+42],
node[v+43])+np.outer(node[v+51], node[v+44])+np.outer(node[v+55], node[v+45])+np.outer(node[v+47],
node[v+46])+np.outer(node[v+46], node[v+47])+np.outer(node[v+49], node[v+48])+np.outer(node[v+48],
node[v+49])+np.outer(node[v+53], node[v+50])
ys_P3=ys5 +np.outer(node[v+44], node[v+51])+np.outer(node[v+41], node[v+52])+np.outer(node[v+50],
node[v+53])+np.outer(node[v+59], node[v+54])+np.outer(node[v+45], node[v+55])+np.outer(node[v+60],
node[v+56])+np.outer(node[v+58], node[v+57])+np.outer(node[v+57], node[v+58])+np.outer(node[v+54],
node[v+59])+np.outer(node[v+56], node[v+60])

#|Z><Z|
zs1 = np.outer(node[6], node[1])+np.outer (node[7], node[2])+np.outer(node[8], node[3])+np.outer (node[5], node[4])+np.outer
(node[4], node[5])+np.outer (node[1], node[6])+np.outer(node[2], node[7])+np.outer(node[3], node[8])
zs2 = zs1 +np.outer(node[21], node[11])+np.outer(node[22], node[12])+np.outer(node[23], node[13])+np.outer(node[24],
node[14])+np.outer(node[25], node[15])+np.outer(node[26], node[16])+np.outer(node[27], node[17])+np.outer(node[28],
node[18])+np.outer(node[29], node[19])+np.outer(node[30], node[20])
zs_P1 = zs2 +np.outer(node[11], node[21])+np.outer(node[12], node[22])+np.outer(node[13], node[23])+np.outer(node[14],
node[24])+np.outer(node[15], node[25])+np.outer(node[16], node[26])+np.outer(node[17], node[27])+np.outer(node[18],
node[28])+np.outer(node[19], node[29])+np.outer(node[20], node[30])
#-------------------------------------------------------------------------------------------------------------------
zs4 =np.outer(node[v+41], node[v+31])+np.outer(node[v+42], node[v+32])+np.outer(node[v+43], node[v+33])+np.outer(node[v+44],
node[v+34])+np.outer(node[v+45], node[v+35])+np.outer(node[v+46], node[v+36])+np.outer(node[v+47],
node[v+37])+np.outer(node[v+48], node[v+38])+np.outer(node[v+49], node[v+39])+np.outer(node[v+50], node[v+40])
zs5 =zs4 +np.outer(node[v+31], node[v+41])+np.outer(node[v+32], node[v+42])+np.outer(node[v+33],
node[v+43])+np.outer(node[v+34], node[v+44])+np.outer(node[v+35], node[v+45])+np.outer(node[v+36],
node[v+46])+np.outer(node[v+37], node[v+47])+np.outer(node[v+38], node[v+48])+np.outer(node[v+39],
node[v+49])+np.outer(node[v+40], node[v+50])
zs_P3 =zs5 +np.outer(node[v+56], node[v+51])+np.outer(node[v+57], node[v+52])+np.outer(node[v+58],
node[v+53])+np.outer(node[v+60], node[v+55])+np.outer(node[v+51], node[v+56])+np.outer(node[v+52],
node[v+57])+np.outer(node[v+53], node[v+58])+np.outer(node[v+55], node[v+60])

# the except four jumping space are(corresponding to |Z><X| and |X><Z|)
node9_17 = np.outer(node[9], node[17])
node17_9 = np.outer(node[17], node[9])
node10_18 = np.outer(node[10], node[18])
node18_10 = np.outer(node[18], node[10])
#-----------
node54_48_P3 = np.outer(node[(v+54)], node[(v+48)])
node48_54_P3 = np.outer(node[(v+48)], node[(v+54)])
node59_60_P3 = np.outer(node[(v+59)], node[(v+60)])
node60_59_P3 = np.outer(node[(v+60)], node[(v+59)])
#----------
node_xz = node17_9 + node18_10 + node48_54_P3 + node60_59_P3
node_zx = node9_17 + node10_18 + node54_48_P3 + node59_60_P3
#-----------------------------------------------------------
# the coin state terms that we need without change (in 3D)
cxx3 = np.outer(coin3[0], coin3[0])
cyy3 = np.outer(coin3[1], coin3[1])
czz3 = np.outer(coin3[2], coin3[2])
# the coin state terms that we need with change (in 3D)
czx3 = np.outer(coin3[2], coin3[0])
cxz3 = np.outer(coin3[0], coin3[2])
#-------------------------------------------------------------------
#the shift terms P1 + P3:
os_P1P3 = os_P1 + os_P3
xs_P3_zx
xs_P1P3 = xs_P1 + xs_P3
ys_P1P3 = ys_P1 + ys_P3
zs_P1P3 = zs_P1 + zs_P3
#---------------------------------------
# the shift operator parts for (3D):
S_hat_X3_P3_zx = np.kron(xs_P3_zx, czx3)
S_hat_X3_P1P3 = np.kron(xs_P1P3, cxx3)
S_hat_Y3_P1P3 = np.kron(ys_P1P3, cyy3)
S_hat_Z3_P1P3 = np.kron(zs_P1P3, czz3)
```

```
S_hat_X3_P2_xx = np.kron(xs_P2_xx, cxx3)
S_hat_X3_P2 = np.kron(xs_P2, czx3)
S_hat_Y3_P2 = np.kron(ys_P2, cyy3)
S_hat_Z3_P2 = np.kron(zs_P2, cxz3)
#-----------------------------------------------------------------
# the changed coin state terms in the C60 (3D)
S_hat_zx3 = np.kron(node_zx, czx3)
S_hat_xz3 = np.kron(node_xz, cxz3)
#-----------------------------------------------------------------
S_hat_X3 = S_hat_X3_P1P3 + S_hat_X3_P2 + S_hat_X3_P3_zx + S_hat_X3_P2_xx
S_hat_Y3 = S_hat_Y3_P1P3 + S_hat_Y3_P2
S_hat_Z3 = S_hat_Z3_P1P3 + S_hat_Z3_P2
#-----------------------------------------------------------------
os = os_P1P3 + os_P2
xs = xs_P1P3 + xs_P3_zx + xs_P2 + xs_P2_xx
ys = ys_P1P3 + ys_P2
zs = zs_P1P3 + zs_P2
#-----------------------------------------
# Finally the Shift operator is:
S_hatQ3 = S_hat_X3 + S_hat_Y3 + S_hat_Z3 + S_hat_xz3 + S_hat_zx3 # quantum case 3D
S_hatC3 = xs + ys + zs + node_xz + node_zx # classical case 3D
#-----------------------------------------
invers_S_hatQ3 = np.linalg.inv(S_hatQ3)
unitarity_of_S_hatQ3 = np.dot(invers_S_hatQ3, S_hatQ3)
print ('unitarity of S_hatQ3 = ')
print (unitarity_of_S_hatQ3)

print ('these results are for W=' + str(W) + ', and R =' + str(R) + '.')
print ('++++++++++++++++++++++++++')
#===== Evolution operator ============================
UG3 = S_hatQ3.dot(np.kron(np.eye(N), G3))
UC3= S_hatC3/3
#======Difine the initial state Psi( zero)=====================
i=1
space = np.zeros(N)
while i<=(5):
space = space + node[i]
i+=1
psi0Q3 = np.kron((space)/np.sqrt(5),(coin3[0]+coin3[1]+coin3[2])/np.sqrt(3.))
psi0C3 =((space))/(5)
#=============================================
yy= T+1
Arri_Prob_G3 = np.zeros(yy)
Arri_Prob_C3 = np.zeros(yy)
aG3 =0
aC3 =0
bG3 = np.zeros(yy)
bC3 = np.zeros(yy)
Norm_psi_t_G3 = np.zeros(yy)
Norm_psi_t_C3 = np.zeros(yy)
#=============================================
t=0
while t<=T:
        if t==0:
                psi_t_G3 = psi0Q3
                psi_t_C3 = psi0C3
        else :
                psi_t_G3 = np.dot(UG3, psi_t_G3)
                psi_t_C3 = np.dot(UC3, psi_t_C3)
        #====Grover 3D=================================
        e3 = ((v+55)*3)
        while e3<((v+60)*3):
                psi_t_G3[e3] = 0
                e3+=1
        #-------------------------------------------
        Norm_psi_t_G3[t] = psi_t_G3.dot(psi_t_G3.conjugate()).real
        Arri_Prob_G3[t] = 1 - Norm_psi_t_G3[t]
        #=========classical case 3D=====================
        psi_t_C3[v+55]=0
        psi_t_C3[v+56]=0
        psi_t_C3[v+57]=0
        psi_t_C3[v+58]=0
        psi_t_C3[v+59]=0
        #---------------------------------
        Norm_psi_t_C3[t]=np.sum(psi_t_C3)
        Arri_Prob_C3[t] = 1 - Norm_psi_t_C3[t]
        t+=1
t-=1
#===== the plotting part ==========================================
minorLocator = AutoMinorLocator()
fig, ax = plt.subplots()
plt.plot(Arri_Prob_G3, 'g-', label='G 3D', markersize=12, linewidth=7)
plt.plot(Arri_Prob_C3, 'k-', label='C 3D', markersize=12, linewidth=7)
```

```
ax.xaxis.set_minor_locator(minorLocator)
ax.tick_params(axis='x', which='major', labelsize=70, width=6, length=20, colors='k')
ax.tick_params(axis='x', which='minor', labelsize=70, width=2, length=20, colors='k')
ax.tick_params(axis='y', which='major', labelsize=70, width=2, length=20, colors='k')
ax.set_xlabel('Steps number', fontsize=65)
ax.set_ylabel('Arrival probability', color='k', fontsize=65)
for tl in ax.get_yticklabels():
tl.set_color('k')
plt.xlim(0, yy)
plt.ylim(0, 1)
plt.title('Arrival probability.ArmChair capped ('+str(Lev)+' Levels),'+str(W)+' width,'+str(R)+'+8 ring, initial state: 5
node(Pentagonal) + equal mixture of all coin states.after '+str(T)+' steps', fontsize=16)
plt.legend(fontsize=20, loc='best', borderaxespad=0.)
figManager = plt.get_current_fig_manager()
figManager.window.showMaximized()
plt.tight_layout()
plt.show()
print ('الحمد لله')
#------the End of the code ----------------------------------------------------------------------------------------
```

المـراجـع:

[1] J. Kempe, Quantum random walks – an introductory overview. Contemp. Phys. **44**(4), 302–327 (2003).

[2] T.M. Cover, J. Thomas, Elements of Information Theory. Wiley, New York (1991)

[3] C. Moore, S. Mertens, The Nature of Computation. Oxford University Press, New York (2011)

[4] W. Feller, An Introduction to Probability Theory and Its Applications, vol. 1, 3rd edn. Wiley, New York (1968)

[5] B.D. Hughes, Random Walks and Random Environments: Random Walks (Vol 1). Clarendon Press, Oxford (1995)

[6] B.D. Hughes, Random Walks and Random Environments: Random Environments (Vol 2). Oxford University Press, Oxford (1996)

[7] A. N. Michael, L. C. Isaac, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge (2000)

[8] R. Portugal, Quantum Walks and Search Algorithms, Quantum Science and Technology. Springer-Verlag New York 2013.

[9] K. Manouchehri, J. Wang, Physical Implementation of Quantum Walks, Quantum Science and Technology. Springer-Verlag Berlin Heidelberg 2014.

[10] N. Satapathy, H. Hagman, M. Zelan, A. Kastberg, and H. Ramachandran, Theoretical investigations of quantum walks by cold atoms in a double optical lattice, Phys. Rev. A **80**, 012302 (2009).

[11] R. Côté, A. Russell, E. E. Eyler, and P. L. Gould, Quantum random with rydberg atoms in an optical lattice, New J. Phys. **8**, 156 (2006).

[12] L. Tarruell, D. Greif, T. Uehlinger, G. Jotzu, and T. Esslinger, Creating, moving and merging dirac points with a fermi gas in a tunable honeycomb lattice, Nature (London) **483**, 302 (2012).

[13] M. Polini, F. Guinea, M. Lewenstein, H. C. Manoharan, and V. Pellegrini, Artificial honeycomb lattices for electrons, atoms and photons, Nat. Nanotechnol. **8**, 625 (2013).

[14] B. C. Travaglione and G. J. Milburn, Implementing the quantum random walk, Phys. Rev. A **65**, 032310 (2002).

[15] J. Kempe, Discrete quantum walks hit exponentially faster, in *ARCO.AT to Fix*, Lecture Notes in Computer Science, (Springer, Berlin, 2003), Vol. **2**764, p. 781.

[16] J. Kempe, Quantum random walks hit exponentially faster, Prob. Theor. Rel. Fields **133**, 215 (2005).

[17] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, Exponential algorithmic speedup by a quantum walk, in *Proceedings of the 35th Annual ACM STOC* (ACM, New York, 2003) pp. 59–68.

[18] S. Bose, Quantum Communication through an Unmodulated Spin Chain, Phys. Rev. Lett. **91**, 207901 (2003).

[19] A. Kay, A review of perfect state transfer and its applications as **6** a constructive tool, Int. J. Quantum Inf. **8**, 641 (2010).

[20] A. Kay, The basics of perfect communication through quantum networks, Phys. Rev. A **84**, 022337 (2011).

[21] P. W. Anderson, Absence of diffusion in certain random lattices, Phys. Rev. **109**, 1492 (1958).

[22] J. P. Keating, N. Linden, J. C. F. Matthews, and A. Winter, Localization and its consequences for quantum walk algorithms and quantum communication, Phys. Rev. A **76**, 012315 (2007).

[23] H. Krovi and T. A. Brun, Hitting time for quantum walks on the hypercube, Phys. Rev. A **73**, 032341 (2006).

[24] M. Bednarska, A. Grudka, P. Kurzynski, T. Łuczak, and A. Wojcik, Quantum walks on cycles, Phys. Lett. A **317**, 21 (2003).

[25] B. Tregenna, W. Flanagan, R. Maile, and V. Kendon, Controlling discrete quantum walks: coins and initial states, New J. Phys. **5**, 83 (2003).

[26] P. R. Dukes, Quantum state revivals in quantum walks on cycles, Results Phys. **4**, 189 (2014).

[27] H. Krovi and T. A. Brun, Quantum walks with infinite hitting times, Phys. Rev. A **74**, 042334 (2006).

[28] H. Krovi and T. A. Brun, Quantum walks on quotient graphs, Phys. Rev. A **75**, 062332 (2007).

[29] Y. Omar, N. Paunkovic, L.S.S. Bose, Quantum walk on a line with two entangled particles. Phys. Rev. A **74**, 042304 (2006)

[30] S.D. Berry, J.B. Wang, Two-particle quantum walks: entanglement and graph isomorphism testing. Phys. Rev. A **83**, 042317 (2011)

[31] M. Stefanak, T. Kiss, I. Jex, B. Mohring, The meeting problem in the quantum walk. J. Phys. A **39**, 14965 (2006)

[32] V. Kendon and B. C. Sanders, Complementarity and quantum walks, Phys. Rev. A **71**, 022307 (2005)

[33] V. Kendon, Decoherence in quantum walks – a review. Math. Struct. Comput. Sci. **17**, 1169 (2007)

[34] V. Kendon, B. Tregenna, QCMC'02: Proceedings of the 6th International Conference on Quantum Communication, Measurement and Computing, p. 463. Rinton Press, (2002).

[35] V. Kendon, B. Tregenna, Decoherence can be useful in quantum walks. Phys. Rev. A **67**, 042315 (2003)

[36] K. S. Novoselov *et al.*, Electric field effect in atomically thin carbon films, Science **306**, 666 (2004).

[37] K. S. Novoselov *et al.*, Two-dimensional gas of massless dirac fermions in graphene, Nature (London) **438**, 197 (2005).

[38] C. T. White and T. N. Todorov, Carbon nanotubes as long ballistic conductors, Lett. Nat. **393**, 240 (1998).

[39] J. W. Mintmire, B. I. Dunlap, and C. T. White, Are Fullerene Tubules Metallic? Phys. Rev. Lett. **68**, 631 (1992).

[40] N. Hamada, S.-i. Sawada, and A. Oshiyama, New OneDimensional Conductors: Graphitic Microtubules, Phys. Rev. Lett. **68**, 1579 (1992).

[41] R. Saito, M. Fujita1, G. Dresselhaus, and M. S. Dresselhaus, Electronic structure of chiral graphene tubules, Appl. Phys. Lett. **60**, 2204 (1992).

[42] M. Dale, J. F. Miller, M. A. Stepney, and S. Trefzer, Evolving carbon nanotube reservoir computers, in *Unconventional Computation and Natural Computation: Proceedings of the 15th International Conference, UCNC 2016, Manchester, United Kingdom, July 2016*, edited by Martyn Amos and Anne Condon (Springer International, Cham, 2016), pp. 49–61.

[43] S. J. Tans, A. R. M. Verschueren, and C. Dekker, Room-temperature transistor based on a single carbon nanotube, Lett. Nat. **393**, 49 (1998).

[44] A. Bachtold, P. Hadley, T. Nakanishi, and C. Dekker, Logic circuits with carbon nanotube transistors, Science **294**, 1317 (2001).

[45] W. J. Yu *et al.*, Adaptive logic circuits with doping-free ambipolar carbon nanotube transistors, Nano Lett. **9**, 1401 (2009).

[46] K. D. Clegg, J. F. Miller, K. Massey, and M. Petty, Travelling salesman problem solved 'in materio' by evolved carbon nanotube device, in *Parallel Problem Solving from Nature–PPSN XIII: Proceedings of the 13th International Conference, Ljubljana, Slovenia, September 2014*, edited by Thomas Bartz-Beielstein, Jurgen Branke, Bogdan Filipic, and Jim Smith (Springer International, Cham, 2014), p. 692.

[47] I. Foulger, S. Gnutzmann, and G. Tanner, Quantum walks and quantum search on graphene lattices, Phys. Rev. A **91**, 062323 (2015).

[48] C. Lyu, L. Yu, and S. Wu, Localization in quantum walks on a honeycomb network, Phys. Rev. A **92**, 052305 (2015).

[49] M. A. Jafarizadeh and R. Sufiani, Investigation of continuous-time quantum walk on root lattice and honeycomb lattice, Physica A: Stat. Mech. Appl. **381**, 116 (2007).

[50] G. Abal, R. Donangelo, F. L. Marquezino, and R. Portugal, Spatial search in a honeycomb network, Math. Struct. Comput. Sci. **20**, 999 (2010).

[51] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, Quantum walks on graphs, in *Proceedings of the 33rd Annual ACM STOC* (ACM, New York, 2001), pp. 50–59.

[52] C. Moore and A. Russell, Quantum walks on the hypercube, in *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM '02)*, edited by J. D. P. Rolim and S. Vadhan (Springer, New York, 2002), pp. 164–178.

[53] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous, One-dimensional quantum walks, in *Proceedings of the 33rd Annual ACM STOC* (ACM, New York, 2001), pp. 60.

[54] G. Grimmett, S. Janson, and P. Scudo, Weak limits for quantum random walks, Phys. Rev. E **69**, 026119 (2004).

[55] A. D. Gottlieb, S. Janson, and P. F. Scudo, Convergence of coined quantum walks in R*d*, Inf. Dimen. Anal. Quantum Probab. Rel. Topics **8**, 129 (2005).

[56] V. M. Kendon and C. Tamon, Perfect state transfer in quantum walks on graphs, J. Comp. Theor. Nanoscience **8**, 422 (2010).

[57] R. J. Angeles-Canul, R. Norton, M. Opperman, C. Paribello, M. Russell, and C. Tamon, Perfect state transfer, integral circulants and join of graphs, Quantum Inf. Comput. **10**, 325 (2010).

[58] G. V Rossum, and F. L. Drake (eds.), Python Software Foundation, https://www.python.org/ (2006).

[59] E Jones, T. Oliphant, P. Peterson, and others; SciPy: Open source scientific tools for Python, http://www.scipy.org/.

[60] D. Ascher, P. F. Dubois, K. Hinsen, J. Hugunin, and T. Oliphant, Numerical Python (1999).

[61] J. D. Hunter, Matplotlib: A 2D graphics environment; Computing In Science & Engineering, Vol. 9.

[62] Virtual NanoLab version 2016.0, QuantumWise A/S, www.quantumwise.com.

[63] H. Bougroura, H. Aissaoui, N. Chancellor, V. Kendon. Quantum-walk transport properties on graphene structures, Phys. Rev. A **94**, 062331 (2016)

[64] F. W. Strauch, Connecting the discrete and continuous-time quantum walks, Phys. Rev. A **74**, 030301 (2006).

Quantum-walk transport properties on graphene structures

Abstract:

We present numerical studies of quantum walks on C_{60} and related graphene structures, to investigate their transport properties. Also known as a honeycomb lattice, the lattice formed by carbon atoms in the graphene phase can be rolled up to form nanotubes of various dimensions. Graphene nanotubes have many important applications, some of which rely on their unusual electrical conductivity and related properties. Quantum walks on graphs provide an abstract setting in which to study such transport properties independent of the other chemical and physical properties of a physical substance. They can thus be used to further the understanding of mechanisms behind such properties. We find that nanotube structures are significantly more efficient in transporting a quantum walk than cycles of equivalent size, provided the symmetry of the structure is respected in how they are used. We find faster transport on zig-zag nanotubes compared to armchair nanotubes, which is unexpected given that for the actual materials the armchair nanotube is metallic, while the zig-zag is semiconducting.

Keywords: quantum information, quantum walk, discrete-time model, transport propriety, graphene structure, Bucky-ball (C_{60}), torus nanotube, capped nanotube,

Marche-quantique Propriétés de transport sur les structures de graphène

Résumé:

Nous avons présenté une étude numérique de la marche quantique dans la molécule C_{60} et des structures de graphène apparentées, pour étudier leurs propriétés de transport. Egalement connu sous le nom de "honeycomb lattice", le treillis formé par des atomes de carbone dans la phase de graphène peut être enroulé pour former des nanotubes de différentes dimensions. Les nanotubes de graphène ont de nombreuses applications importantes, dont certaines dépendent de leur conductivité électrique inhabituelle et des propriétés connexes. La marche quantique sur les graphes fournit un cadre abstrait pour étudier ces propriétés de transport indépendamment des autres propriétés chimiques et physiques d'une substance physique. Ils peuvent ainsi être utilisés pour approfondir la compréhension des mécanismes derrière ces propriétés. Nous constatons que les structures de nanotubes sont significativement plus efficaces dans le transport d'une marche quantique que les cycles de taille équivalente, à condition que la symétrie de la structure est respectée dans la façon dont ils sont utilisés. Nous trouvons un transport plus rapide sur les nanotubes en zigzag par rapport aux nanotubes en armchair, ce qui est inattendu étant donné que pour les matériaux réels, le nanotube d'armchair est métallique, tandis que le zigzag est semi-conducteur.

Mots-clés : information quantique, marche quantique, discrète-temps model, transport propriété, graphène structure, Bucky-ball (C_{60}), torus nanotube, capped nanotube,

الملخص:

قدّمنا هنا في هذه الدارسة. معالجة رقمية تحاكي المشي الكمي في بنية جزيء الكربون C_{60} و بنى الجرافين المتعلقة بها، من أجل اختبار خاصية الانتقال. شبكة الجرافين المستوية، و المعروفة أيضا تحت مسمى "شبكة عسل النحل"، هي شبكة مكونة من ذرات الكربون في شكل سداسيات يمكن لَفُهَا لتشكيل أنابيب النانو بأحجام مختلفة. هذه الأنابيب لها العديد من التطبيقات الهامة، تعتمد بعضها على ناقليتها الكهربائية الغير مألوفة و الخواص المتعلقة بها. إن فكرة المشي العشوائي الكمي تطبق عموما عبر بنى مجردة (المخطاط)، مما يعني أن نتائجنا ستكون مستقلة عن مختلف الخصائص الكيميائية و الفيزيائية لمهية مكونات البنى المدروسة. و بالتالي سيعزز هذا أكثر في فهم حقيقة الآلية المسؤولة عن خاصية الانتقال. لقد وجدنا أنا بنى أنابيب النانو تكون و بشكل ملحوظ أكثر كفاءة في اختبار خاصية التنقل مقارنة ببنى مثل الحلقات من نفس الحجم، شريطة أن نحسن اختيار و استعمال التناظرات المتوفرة في بنى الجرافين. كما وجدنا أن التنقل يكون أسرع في أنابيب النانو من النوع zigzag مقارنة بنظريتها من نوع armchair، و هي نتيجة غير منتظرة كون علم المواد يصنف اليوم أنابيب armchair من جنس المعادن و أنابيب zigzag من جنس أنصاف النواقل.

الكلمات المفتاحية: المعلوماتية الكمية، المشي الكمي، النموذج الغير مستمر، خاصية الانتقال، بنى الجرافين، جزيء الكربون C_{60}، أنابيب النانو الحلقية، أنابيب النانو المقببة.