CHƯƠNG TRÌNH TÍNH GFR

1. GIỚI THIỆU

   Chương trình xử lý số liệu thu được để tính GFR bằng kỹ thuật Two-point Patlak plot. Chương trình được viết bằng ngôn ngữ C++.

2. PHÂN TÍCH CHƯƠNG TRÌNH

   2.1. TỔ CHỨC DỮ LIỆU

- Mỗi bệnh nhân có hai folder dữ liệu ứng với thận trái và thận phải. Mỗi folder gồm bốn file dữ liệu (ở đây ta lấy ví dụ folder ứng với thận trái của bệnh nhân Phạm Tiến Hoàng):
  - File dữ liệu của động mạch chủ:
    - Gồm n hàng ứng với n lát cắt, được sắp xếp từ trên xuống theo thứ tự: thì chụp khởi động sau tiêm thuốc nhanh, thì chụp động mạch, thì chụp động thứ nhất, thì chụp động thứ hai, thì chụp nhu mô.
    - Mỗi hàng chứa thời điểm và tỷ trọng của lát cắt.
    - Tên file: Left_Pham_Tien_Hoang
  - File dữ liệu K thì không thuốc:
    - Gồm n hàng ứng với n lát cắt. Mỗi hàng chứa tỷ trọng và diện tích của lát cắt.
    - Tên file: “Kunenhanced_Left_Pham_Tien_Hoang”.
  - File dữ liệu K thì động mạch:
    - Gồm n hàng ứng với n lát cắt. Mỗi hàng chứa tỷ trọng và diện tích của lát cắt.
    - Tên file: “Karterial_Left_Pham_Tien_Hoang”.
  - File dữ liệu K thì nhu mô:
    - Gồm n hàng ứng với n lát cắt. Mỗi hàng chứa tỷ trọng và diện tích của lát cắt.
    - Tên file: “Kparenchymal_Left_Pham_Tien_Hoang”.

- Cách nhập tên bệnh nhân lúc bắt đầu chương trình: Ví dụ với thận trái bệnh nhân Phạm Tiến Hoàng, ta sẽ nhập: “Left_Pham_Tien_Hoang”.

2.2. CÁCH CHƯƠNG TRÌNH XỬ LÝ SỐ LIỆU

2.2.1. Xử lý số liệu động mạch chủ

Sau khi nhập tên bệnh nhân, chương trình sẽ mở file dữ liệu động mạch chủ và đọc dữ liệu. Sau khi nhập tỷ trọng ở thì không thuốc, chương trình sẽ tính tỷ trọng thực của mỗi lát cắt bằng cách lấy tỷ trọng của mỗi lát cắt trừ đi tỷ trọng ở thì không thuốc. Ta được thời điểm và tỷ trọng thực tương ứng của mỗi lát cắt.

```
cout << "Enter patient name: ";
cin >> name;
inFile.open(name);
string s;
int number_of_scan = 0;
while(getline(inFile, s)) {
   number_of_scan ++;
}
inFile.clear();
inFile.seekg(0);
int number_of_data = number_of_scan + 1;
double rawdata[number_of_data][2];
for (int i=1; i < number_of_data; i++) {
   inFile >> rawdata[i][0] >> rawdata[i][1];
}
double densityUn;
cout << "Unenhanced density: ";
cin >> densityUn;
```

```
double data[number_of_data][2];
data[0][0] = 0;
data[0][1] = 0;
for (int i=1; i < number_of_data; i++) {
   data[i][0] = rawdata[i][0];
   data[i][1] = rawdata[i][1] - densityUn;
}
```

Tiếp theo, chương trình sẽ duyệt dữ liệu vừa tính được, những tỷ trọng thực có cùng một thời điểm, sẽ được lấy trung bình và gán giá trị tỷ trọng trung bình này cho thời điểm đó. Ta được thời điểm và tỷ trọng thực trung bình ứng với thời điểm đó.

```
 int count=1;
for (int i=0; i < number_of_data-1; i++) {
   if (data[i][0] != data[i+1][0]) {
      count++;
   }
}
double time[count];
int j=1;
time[0] = data[0][0];
for (int i=0; i < number_of_data-1; i++) {
   if (data[i][0] != data[i+1][0]) {
      time[j] = data[i+1][0];
      j++;
   }
}
double density[count];
density[0] = data[0][1];
```

```
double sum[count];
double dem[count];
for (int i=1; i < count; i++) {
   sum[i]=0;
   dem[i]=0;
 for (int k=1; k < number_of_data; k++) {
      if (data[k][0] == time[i]) {
         sum[i] += data[k][1];
         dem[i] ++;
      }
   }
   density[i] = sum[i] / dem[i];
}
```

2.2.2. Lấy trung bình tỷ trọng thực ở thì động mạch và thì nhu mô

Sau khi nhập thời điểm của lát cắt đầu và lát cắt cuối của các thì chụp động mạch và thì chụp nhu mô, chương trình sẽ lấy trung bình tỷ trọng thực của các lát cắt của thì chụp động mạch và tỷ trọng thực của các lát cắt của thì chụp nhu mô.

```
double ArBgn, ArEnd, PaBgn, PaEnd;
int ArBgnIndex = 0, ArEndIndex = 0, PaBgnIndex = 0, PaEndIndex = 0;
cout << "Time of first Arterial phase scan: ";
cin >> ArBgn;
cout << "Time of last Arterial phase scan: ";
cin >> ArEnd;
cout << "Time of first Parenchymal phase scan: ";
cin >> PaBgn;
cout << "Time of last Early Parenchymal phase scan: ";
cin >> PaEnd;
```

```
for (int i=0; i < count; i++){
    if (time[i] == ArBgn) {
        ArBgnIndex = i;
    }
    else if (time[i] == ArEnd) {
        ArEndIndex = i;
    }
    else if (time[i] == PaBgn) {
        PaBgnIndex = i;
    }
    else if (time[i] == PaEnd) {
        PaEndIndex = i;
    }
}
double ArIntDens=0, PaIntDens=0, sum1=0, sum2=0, count1=0, count2=0, intsum1=0, intsum2=0;
double timeNew[count], densityNew[count];
for (int i=0; i < count; i++){
    if ( (time[i] < ArBgn) || ((time[i] > ArEnd) && (time[i] < PaBgn)) || (time[i] > PaEnd) ) {
        timeNew[i] = time[i];
    }
    else if ((time[i] >= ArBgn) && (time[i] <= ArEnd)) {
        sum1 += density[i];
        count1++;
        timeNew[i] = time[i];
    }
```

```
        else if ((time[i] >= PaBgn) && (time[i] <= PaEnd)) {
            sum2 += density[i];
            count2++;
            timeNew[i] = time[i];
        }
    }
    ArIntDens = sum1 / count1;
    PaIntDens = sum2 / count2;
    for (int i=0; i < count; i++){
        if ( ((time[i] < ArBgn) || (time[i] > ArEnd)) && ((time[i] < PaBgn) || (time[i]
> PaEnd)) ) {
            densityNew[i] = density[i];
        }
        else if ((time[i] >= ArBgn) && (time[i] <= ArEnd)) {
            densityNew[i] = ArIntDens;
        }
        else if ((time[i] >= PaBgn) && (time[i] <= PaEnd)) {
            densityNew[i] = PaIntDens;
        }
    }
```

2.2.3. Xây dựng b(t)

Với hai điểm dữ liệu liên tiếp, chương trình sẽ viết phương trình đường thẳng đi qua hai điểm đó. Từ đó xây dựng được hàm tỷ trọng theo thời gian.

```
    double a[count-1], b[count-1];
    for (int i = 0; i < count-1; i++){
        a[i] = (densityNew[i+1] - densityNew[i]) / (timeNew[i+1] - timeNew[i]);
        b[i] = -a[i] * timeNew[i] + densityNew[i];
```

```
}
cout << "\n";
cout << "Linear interpolation: \n";
cout << "b(t) = \n";
for (int i = 0; i < ArBgnIndex; i++){
    cout << a[i] << "t + " << b[i] << " if " << timeNew[i] << " <= t < " << timeNew[i+1] << "\n";
}
cout << ArIntDens << " if " << ArBgn << " <= t < " << ArEnd << "\n";
for (int i = ArEndIndex; i < PaBgnIndex; i++){
    cout << a[i] << "t + " << b[i] << " if " << timeNew[i] << " <= t < " << timeNew[i+1] << "\n";
}
cout << PaIntDens << " if " << PaBgn << " <= t < " << PaEnd << "\n";
for (int i = PaEndIndex; i < count-1; i++){
    cout << a[i] << "t + " << b[i] << " if " << timeNew[i] << " <= t < " << timeNew[i+1] << "\n";
}
```

2.2.4. Tính tích phân

Ta định nghĩa thời điểm t1 và t2 lần lượt là thời điểm chính giữa của thì chụp động mạch và thì chụp nhu mô. Chương trình sẽ xác định t1, t2, b(t1), b(t2), và tính tích phân của hàm tỷ trọng từ thời điểm 0 đến thời điểm t1 (t2) bằng cách tính diện tích có dấu phần bị giới hạn bởi đồ thị hàm tỷ trọng từ thời điểm 0 đến thời điểm t1 (t2).

```
cout << "t1 = " << t1 << "\n";
cout << "t2 = " << t2 << "\n \n";
cout << "b(t1) = " << ArIntDens << "\n";
cout << "b(t2) = " << PaIntDens << "\n \n";
/////// Calculate integral from t=0 to t=t1
```

```
for (int i = 0; i < ArBgnIndex; i++){
    intsum1 += (densityNew[i] + densityNew[i+1]) * (timeNew[i+1] - timeNew[i]) * 0.5;
}
intsum1 = intsum1 + (t1 - ArBgn)*ArIntDens;
cout << "The integral of b(t) from t0 to t1: " << intsum1 << "\n";
/////// Calculate integral from t=0 to t=t2
for (int i = ArEndIndex; i < PaBgnIndex; i++){
    intsum2 += (densityNew[i] + densityNew[i+1]) * (timeNew[i+1] - timeNew[i]) * 0.5;
}
  intsum2 = intsum2 + (ArEnd - t1)*ArIntDens + (t2 - PaBgn)*PaIntDens + intsum1;
cout << "The integral of b(t) from t0 to t2: " << intsum2 << "\n";
cout << "The integral of b(t) from t1 to t2: " << intsum2 - intsum1 << "\n \n";
```

2.2.5. Xử lý số liệu của các file K và tính volumn data, thể tích nhu mô thận

Sau khi nhập độ dày lát cắt, chương trình sẽ đọc ba file dữ liệu K và tính K(t1), K(t2), volumn data ở các thì, thể tích nhu mô thận ở các thì và thể tích nhu mô thận trung bình.

```
double thickness;
cout << "Slice thickness: ";
cin >> thickness;
//////// K unenhanced
string name1 = "Kunenhanced_" + name;
KunFile.open(name1);
string s1;
int number_of_scan1 = 0;
```

```
while(getline(KunFile, s1)) {
    number_of_scan1 ++;
}
KunFile.clear();
KunFile.seekg(0);
double rawdata1[number_of_scan1][2]; // rawdata1[i][0]: density; rawdata[i][1]:
area
for (int i=0; i < number_of_scan1; i++) {
    KunFile >> rawdata1[i][0] >> rawdata1[i][1];
}

double KUn[number_of_scan1], KUnSum=0;
for (int i=0; i < number_of_scan1; i++) {
    KUn[i] = rawdata1[i][0] * rawdata1[i][1] * thickness;
}
cout << "K unenhanced \n";
for (int i=0; i < number_of_scan1; i++) {
    cout << KUn[i] << "\n";
}
for (int i=0; i < number_of_scan1; i++) {
    KUnSum = KUnSum + KUn[i];
}
//////// K arterial
string name2 = "Karterial_" + name;
KarFile.open(name2);
string s2;
int number_of_scan2 = 0;
```

```
while(getline(KarFile, s2)) {
   number_of_scan2 ++;
}
KarFile.clear();
KarFile.seekg(0);

double rawdata2[number_of_scan2][2];
for (int i=0; i < number_of_scan2; i++) {
   KarFile >> rawdata2[i][0] >> rawdata2[i][1];
}
double KAr[number_of_scan2], KArSum=0;
for (int i=0; i < number_of_scan2; i++) {
   KAr[i] = rawdata2[i][0] * rawdata2[i][1] * thickness;
}
cout << "\n";
cout << "K Arterial \n";
for (int i=0; i < number_of_scan2; i++) {
   cout << KAr[i] << "\n";
}
for (int i=0; i < number_of_scan2; i++) {
   KArSum = KArSum + KAr[i];
}
//////// K Parenchymal
string name3 = "Kparenchymal_" + name;
KpaFile.open(name3);
string s3;
int number_of_scan3 = 0;
```

```
while(getline(KpaFile, s3)) {
   number_of_scan3 ++;
}
KpaFile.clear();
KpaFile.seekg(0);
double rawdata3[number_of_scan3][2];
for (int i=0; i < number_of_scan3; i++) {
   KpaFile >> rawdata3[i][0] >> rawdata3[i][1];
}
double KPa[number_of_scan3], KPaSum=0;
for (int i=0; i < number_of_scan3; i++) {
   KPa[i] = rawdata3[i][0] * rawdata3[i][1] * thickness;
}
cout << "\n";
cout << "K Parenchymal \n";
for (int i=0; i < number_of_scan3; i++) {
   cout << KPa[i] << "\n";
}
for (int i=0; i < number_of_scan3; i++) {
   KPaSum = KPaSum + KPa[i];
}
cout << "\n";

////////// Volumn data
cout << "Parenchymal Volumn Data Unenhanced: " << KUnSum << "\n";
cout << "Parenchymal Volumn Data in Arterial Phase: " << KArSum << "\n";
```

```
cout << "Parenchymal Volumn Data in Parenchymal Phase: " << KPaSum << "\n \n";
////////////// K(t1), K(t2)
double Kt1 = KArSum - KUnSum;
double Kt2 = KPaSum - KUnSum;
cout << "K(t1) and K(t2): \n";
cout << "K(t1): " << Kt1 << "\n";
cout << "K(t2): " << Kt2 << "\n \n";
////////////// Volumn
double VolUn = 0, VolAr = 0, VolPa = 0;
for (int i=0; i < number_of_scan1; i++) {
   VolUn = VolUn + rawdata1[i][1] * thickness;
}
for (int i=0; i < number_of_scan2; i++) {
   VolAr = VolAr + rawdata2[i][1] * thickness;
}
for (int i=0; i < number_of_scan3; i++) {
   VolPa = VolPa + rawdata3[i][1] * thickness;
}

cout << "Parenchymal Volumn Unenhanced: " << VolUn << " mm^3 \n";
cout << "Parenchymal Volumn in Arterial Phase: " << VolAr << " mm^3 \n";
cout << "Parenchymal Volumn in Parenchymal Phase: " << VolPa << " mm^3 \n";
cout << "Mean Parenchymal Volumn: " << (VolUn + VolAr + VolPa) / 3 << " mm^3 \n \n";
```

2.2.6. Tính c2

Từ các giá trị đã tính được ở trên, chương trình sẽ tính c2 từ công thức đã có.

```
double c2 = (Kt2 - Kt1*(PaIntDens/ArIntDens)) / (intsum2 - (PaIntDens/ArIntDens)*intsum1);
cout << "c2 = " << c2 << "\n";
```

2.2.7. Tính GFR

Từ các giá trị đã tính được ở trên và Hct được nhập vào, chương trình sẽ tính GFR từ công thức đã có. Kết quả được tự động nhân với 0.06 để đổi đơn vị từ mm^3/sec sang mL/min.

```
double Hct;
cout << "Hematocrit level: ";
cin >> Hct;
cout << "GFR(CT) = " << (1-Hct)*c2*0.06 << " mL/min \n";
```